Model Reconstruction Using POD Method for Gray-box Fault Detection[1,2]

Han G. Park, Michail Zak
Jet Propulsion Laboratory
4800 Oak Grove Drive
Pasadena, CA 91109
Han.G.Park@jpl.nasa.gov

Abstract—This paper describes using Proper Orthogonal Decomposition (POD) method to create low-order dynamical models for the Model Filter component of Beacon-based Exception Analysis for Multi-missions (BEAM). The POD modeling procedure is described, and its usefulness in creating simple low-order dynamical models of a complex system. The POD procedure will be shown on an example problem of a Burgers' Equation. It will be demonstrated that stable low-order dynamical models can be created even in the presence of noise commonly found in experimental data.

TABLE OF CONTENTS

1. INTRODUCTION
2. BEAM OVERVIEW
3. MODEL FILTER
4. PROPER ORTHOGONAL DECOMPOSITION
5. GALERKIN PROJECTION
6. EXAMPLE – BURGERS' EQUATION
7. RESULTS
8. CONCLUSION

1. INTRODUCTION

The Proper Orthogonal Decomposition (POD) is a method for creating low-order dynamical models for complex system using either empirical or simulation data. This method has been successfully applied to many complex systems, including complex fluid flows such as those encountered in turbulence. [5],[7] The method has been viewed as a viable method for reducing computational complexity by generating simple models that can be used for control and simulation.

This paper describes applying POD method to create low-order dynamical models for the Model Filter component of Beacon-based Exception Analysis for Multi-missions (BEAM). BEAM is an end-to-end method of data analysis intended for real-time (on-board) or non-real-time anomaly detection and characterization. The Model Filter component, also know as the "gray-box," serves to filter the data using a deterministic model of system, i.e., "white box," and generate a residual which is then described by a stochastic model, i.e., "black-box."

In order to fully implement the "gray-box" fault detector, a deterministic model is necessarily. In many cases, however, a detailed model of the system may not exist, or the model may be too complex to implement for real-time applications. A simplified model that accurately describes the system is desired. The POD method is attractive method for creating simplified models for dynamical systems. It offers the advantage of reducing complex system models to a simple low-order dynamic model. The model can be created either from simulation or empirical data of the complex system. The POD method maximizes a priori physical knowledge about the system by projecting empirical measurement derived modes onto a physical model of the system.

In this paper, the BEAM method of fault detection and analysis will be introduced. The gray-box component will be discussed in further detail to motivate the need for creation of a simplified dynamical model of the system from empirical data. Then the POD technique will be introduced as well as the steps needed to create a simple low-order dynamical model of a complex system. As a demonstration, the POD procedure will be shown on the Burgers' Equation. It will be demonstrated that the POD method can be used to construct a high fidelity low-order dynamical model even in the presence of noise in the data.

2. BEAM OVERVIEW

BEAM is a complete data analysis system for real-time or off-line fault detection and characterization at the signal-level. While the originally intended for generic system analysis on-board deep space probes and other highly automated systems, the compact and modular nature of its

[1] 0-7803-7231-X/01/$10.00/© 2003 IEEE
[2] IEEEAC paper #008, Updated Sept 27, 2001

subroutines naturally lends itself to either ground or flight deployment.

At the simplest level of abstraction, BEAM is software, which takes data as input and reports fault status as output. Implementation of this software dependents on the application, but a typical application would have a system with a number of individual components, each of which reports health or performance data to a local computer. To accommodate such a wide range of possibilities, the computational engine of BEAM itself is highly adaptable with respect to subsystem size and complexity.

For each single component or subsystem, we can expect to receive four types of data:

1. Discrete status variables changing in time – modes, switch positions, health bits, etc. – from sensors or software

2. Real-valued sensor data varying at fixed rates – performance sensors or dedicated diagnostic sensors

3. Command information – typically discrete as in 1.

4. Fixed parameters – varying only when commanded to change but containing important state information

These types of data are all valuable but used in different ways. Status variables and commands are useful to a symbolic model. Commands and fixed parameters are used in a physical system model while the time-varying sensor data are used in signal processing components. An optimal strategy must take each of these into account and produce a single, unified decision. In order to study each and combine results, BEAM has the following architecture. (Figure 1)

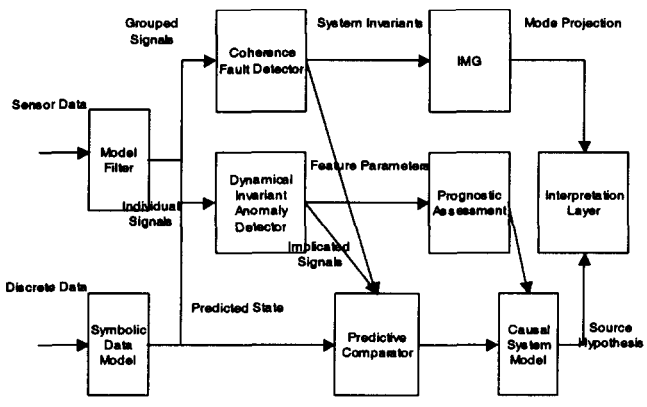

Figure 1. Top-level BEAM architecture.

A few notes about the architecture are in order before we consider the individual descriptions of its components. Specifically, we should consider the data flow, which is slightly complicated:

1. Fixed parameters and command information are input to the specific system modules (if any). These are contained in the Symbolic components (Symbolic Data Model, Predictive Comparator, Causal System Model, and Interpretation Layer) and the Physical (Numeric) components. These data will not be propagated further and influence other components only through the model outputs.

2. Discrete variables will only be propagated through the symbolic components. The symbolic components support the signal processing components by providing a mode determination from the discrete data.

3. Time-varying quantities are separated into two groups as part of the training process. Specifically, signals with high degrees of correlation to others, or those not expected to uniquely indicate a severe fault, are only passed to the Coherence Fault Detector. Signals that may uniquely indicate a fault, along with those already flagged as faulty by the coherence analysis, require additional processing and are also passed through the Dynamical Invariant Anomaly Detector.

4. The split between time varying signals described in 3. is a computational efficiency consideration and reflects general philosophy of operation, but is not essential. Given adequate resources, there is nothing preventing all time-varying signals from being sent to both types of signal analysis at all times.

Physical (Numeric) Models

1. Model Filter (MF): Receives sensor or other quantitative data, conditions it in terms of synchronicity or drop-outs, and combines results with physics model (simulation) predictions.

2. Coherence Detector (CD): Receives multiple conditioned quantitative signals and performs anomaly detection using cross-signal statistical features.

3. Dynamical Invariant Anomaly Detector (DIAD): Receives a single quantitative signal one at a time and performs anomaly detection using a parametric estimate of the residuals.

4. Informed Maintenance Grid (IMG): Combines outputs from the Coherence Detector over long operating periods to detect subthreshold degradation and predict functional failures.

5. Prognostic Assessment (PA): Uses the parametric signal estimates from DIAD to forecast future signal values and identify potential signal faults.

Symbolic Models

6. Symbolic Data Model (SDM): Receives discrete data and constructs an internal state estimate of the system. It detects discrete signal mismatches (explicit faults) and identifies system mode for use by other components.

7. Predictive Comparator (PC): Combines signal implications from the CD and DIAD as well as discrete reports from SDM in an attempt to unify results from the signal-based and symbolic reasoning components.

8. Causal System Model (CSM): Backtracks implications along the system structure to reduce the complexity of fault reports. This is a simplistic form of diagnosis.

9. Interpretation Layer (IL): Fuses results from different components and constructs a final report. It serves as an interface between BEAM and other components.

Specific instantiations of BEAM may omit some of these components depending on the particular characteristics of each system, such as sensor types, cost to maintain, characteristic time scales, and availability of model or training information. For example, only the Dynamical Invariant Anomaly Detector was implemented for Space Shuttle Main Engine (SSME) analysis due to the transient nature of SSME faults. [3]

3. MODEL FILTER

In the BEAM architecture, the Model Filter (MF) and the Dynamical Invariant Anomaly Detector (DIAD) form the approach called the "gray-box" method. [2] It is called a "gray-box" because it incorporates both a "black-box," i.e., stochastic model and a "white-box," i.e., deterministic model. It is a hybrid model incorporating elements from residual-based methods and parametric-estimation methods. It is similar to adaptive-threshold methods in that a residual is generated without any regard for robust residual generation. However, instead of examining the amplitude of the residual as in the case of the adaptive threshold methods, the structure, i.e. model parameters, of the residual is examined. The residual generation is our "white-box." The residual is modeled using techniques similar to the parametric estimation methods. The method is distinct from the standard parametric estimation method in that the system identification is carried out on the residual, not the system variables directly. The residual is parameterized, not the full system. In our terminology, the parameter estimation method is a "black-box."

A block diagram of the gray-box method is shown in Figure 2. After filtering the deterministic components using the model of the system, the residual is separated into its linear, non-linear, and noise components and is fitted to stochastic models. The parameters to the linear, non-linear, and noise models completely describe the residual. The gray-box has several advantages. First, the full model is employed rather than only the model structure as in the case of standard parametric estimation methods. Thus the gray-box takes full advantage of the information about system. Second, the gray-box method can be made robust to modeling errors which can be taken care of during residual modeling. The model of the residual can also describe many unmodeled phenomena in the original model. Finally, the method is applicable to both linear and non-linear systems.

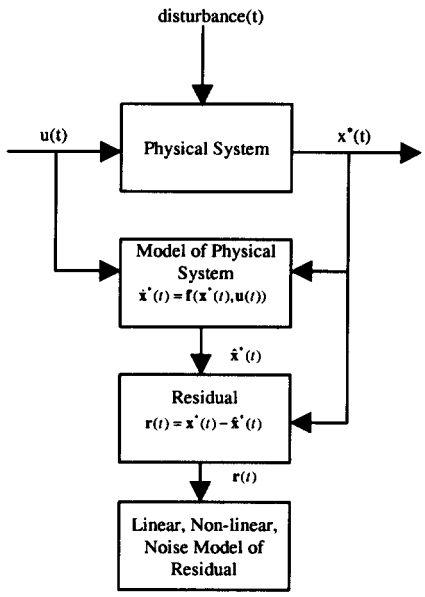

Figure 2. Block diagram of the gray-box method.

The residual generation is as follows. Let us assume that the theoretical model is represented by a system of differential equations:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) + \mathbf{y}(t), \qquad (1)$$

where $\mathbf{x}(t)$ is the state variable vector, $\mathbf{u}(t)$ is the known input, and $\mathbf{f}$ is the known theoretical relationship following from conservation laws of mechanics, thermodynamics, etc. The last term, $\mathbf{y}(t)$, represents components which lack theoretical descriptions, are too complex, or are the result of modeling errors. These can include sensor noise, unknown input to the system, friction in bearings, material viscosity, and other secondary effects such as torsional oscillations of the shaft, flutter in the turbine and compressor blades, incomplete fuel combustion, etc.

The estimate of the system is accomplished by substituting the observed sensor data for the evolution of the state variables, $\mathbf{x}^*(t)$, and input, $\mathbf{u}(t)$. Hence:

$$\dot{\mathbf{x}}^*(t) = \mathbf{f}(\mathbf{x}^*(t), \mathbf{u}(t)). \qquad (2)$$

The residual,

$$\mathbf{r}(t) = \mathbf{x}^*(t) - \hat{\mathbf{x}}^*(t), \qquad (3)$$

is generated by subtracting the solution of Eq. 2, $\hat{\mathbf{x}}^*(t)$, which is generated by using the observed state variables, $\mathbf{x}^*(t)$, from the solution of Eq. 1. Hence the original theoretical model is the filter.

In our gray-box technique, the residual is described by its dynamical invariants comprised of parameters that describe the linear, non-linear, and noise components. The linear component is described by an Auto-Regressive Transfer (ARX) function; the non-linear component, by a time-delay feed-forward neural network with sigmoidal transfer function; and the noise component, by a Markov chain process. The parameters, i.e., dynamical invariants, are a robust measure of the dynamical behavior of the system under observation and provide an easy method for classifying the data as nominal or anomalous by examining the direction of vector.

To take full advantage of the gray-box method, a good deterministic model is required, i.e., **f**. A good deterministic (physical) model of the system is crucial for robust residual generation for maximizing the accuracy of fault detection. However, in many cases, a detailed model of the system does not exist, or the model may be too complex to implement for real-time application. A reliable method for generating a simplified model that accurately describes the system is required. In such case, the POD is an ideal method for creating simplified models for dynamical systems. It offers the advantage of reducing complex system models to a simple low-order dynamic model for real-time applications as well as generating models from empirical data when the physics of the system is known only partially. In the following sections, the POD technique will be illustrated. It will be demonstrated on an example problem of a Burgers' Equation in which a complex nonlinear partial differential equation is reduced to a set of ordinary differential equations.

4. Proper Orthogonal Decomposition

Proper Orthogonal Decomposition (POD) modeling is a method of creating low-dimensional approximate descriptions of high-dimensional processes. POD modeling requires two steps. The first step is to extract the "mode shapes" or basis functions from experimental data or detailed simulations of high-dimensional systems. This step is also known as Principal Component Analysis (PCA), Karhunen-Loeve Decomposition (CLD), or the Singular Value Decomposition (SVD). The second step involves projection of the basis functions to a low-dimensional dynamical model using the Galerkin method.

In the mode extraction step of POD, we wish to approximate a function $u(x,t)$ over some domain of interest as a finite sum in the form of separation of variables:

$$u(x,t) \approx \sum_{k=1}^{M} a_k(t)\varphi_k(x), \qquad (4)$$

where x, t are the spatial coordinate (possibly a vector) and time, respectively. The representation of Eq. 4 is not unique. If the domain of x is a bounded on interval X, then the $\varphi_k(x)$ can be a Fourier series, Legendre polynomials, Chevyshev polynomials, etc. In the case of POD, the $\varphi_k(x)$ are chosen through modes extracted from SVD analysis.

If basis functions are orthonormal:

$$\int_X \varphi_{k1}(x)\varphi_{k2}(x)dx = \{\ 1 \text{ if } k_1 = k_2,\ 0 \text{ if } k_1 \neq k_2\ \}, \qquad (5)$$

then

$$a_k(t) = \int_X z(x,t)\varphi_k(x)dx, \qquad (6)$$

in the least square sense.

In the case of discrete data, we evaluate $u(x,t)$ at N instants of time, thereby taking take N sets of m simultaneous measurements at these m locations in x. We arrange our data in an N x m matrix A where the elements of A_{ij} is the measurement from the jth probe taken at the ith instance of time.

$$A = U\Sigma V^T, \qquad (7)$$

where U is an N x N orthogonal matrix, V is an m x m orthogonal matrix, and Σ is an N x m diagonal matrix. The diagonal elements Σ_{ii} consist of nonnegative numbers σ_i, which are arranged in decreasing order. The σ_i are the singular values of A and are unique.

$$A = QV^T = \sum_{k=1}^{m} q_k v_k^T, \qquad (8)$$

where $Q = U\Sigma$, q_k be the kth column of Q and v_k be the kth column of V. In terms of Eq. (4), q_k represents $a_k(t)$, and v_k^T represents $\varphi_k(x)$.

For POD modeling, a lower rank approximation to A is desirable in order to reduce the order of the equation. A lower k rank approximation of A may be obtained by setting the $\sigma_{K+1} = \sigma_{K+2} = \sigma_{K+3} = \sigma_{K+4} = \ldots = \sigma_m = 0$. The lower rank approximation is given by:

$$A_k = U\Sigma_k V^T, \qquad (9)$$

and

$$Q = A_k V . \qquad (10)$$

The cut-off maybe determined through various methods, including simple inspection, statistical methods, time history, etc. A full discussion of this topic can be found in [4].

5. Galerkin Projection

For a system governed by a partial differential equation governed by the form:

$$\frac{\partial u}{\partial t} = D_\lambda(u), \qquad (11)$$

where u(x,t) is some function of space and time and D_λ is a nonlinear differential operator that depends on parameter λ. For example, $\lambda = (M, \mathrm{Re})$, where M is the Mach number, and Re is the Reynolds number in fluid mechanics.

To compute the approximate solution to (11), we project the equations onto a finite-dimensional subspace. The finite space in this case is the space spanned by the POD modes as shown in the previous section. By taking the inner product with Eq. 4:

$$\dot{a}_k = \left\langle D_\lambda(u), \varphi_k \right\rangle, \qquad (12)$$

we obtain a set of ordinary differential equation for the time dependent coefficients, $a_k(t)$. Please refer to [5] for more information on Galerkin projection.

6. Example – Burgers' Equation

The POD procedure will be shown on the Burgers' Equation as given by:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} = \mu \frac{\partial^2 u}{\partial x^2}, \qquad (13)$$

where $u(x,t)$ is the wave propagation speed, and μ is the diffusion coefficient. Burgers' Equation is a nonlinear partial differential equation that can describe commonly encountered phenomenon such as gas dynamics, flood waves, glaciers, car traffic, etc. [6] In the case of automobile traffic, it can describe velocity of an automobile wherein the driver adjusts his or her speed according to the traffic density. It can also describe flow of information packets through a network. Similar to automobiles, the rate at which packets move through switches can be simulated via the Burgers' Equation.

The simulation of Burgers' Equation with μ = 0.05 using Spectral element method of 97 elements is shown in Figure 3. The initial conditions were:

$$u(x,0) = sin(x) \quad (0 < x < 2\pi) \qquad (14)$$

The results for time t = 0 to t = 1.0 are shown in blue. As time advances, the wave front becomes steeper and steeper. In time, a shockwave will develop when the wave front reaches vertical. The black lines represent simulated empirical data where 10% uncorrelated Gaussian noise were added.

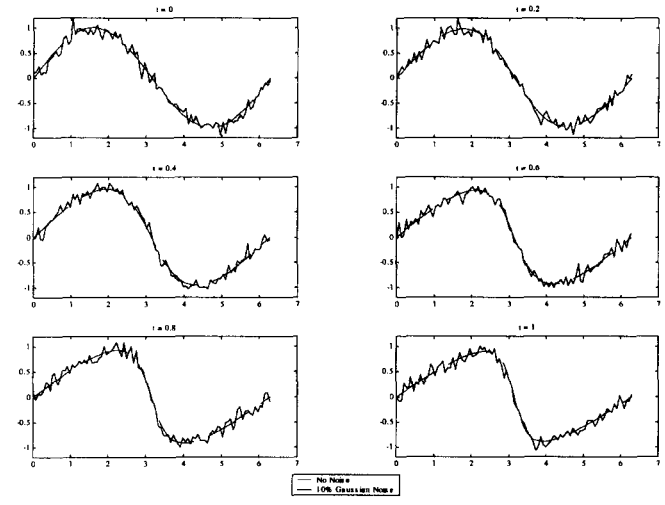

Figure 3. Burgers' Equation simulation with initial condition $u(x,0) = sin(x)$ over $(0, 2\pi)$. The results are shown in blue. The results with 10% uncorrelated Gaussian noise are shown in black.

7. Results

The extracted POD modes from the simulation of Burgers' Equation (Figure 3) are shown in Figure 4. The first six POD modes are shown. It is evident that the modes are symmetric and odd functions about the center. This is due to the nature of Burgers' Equation which is anti-symmetric about π. The higher modes are comprised of progressively higher frequency components. The contribution of each mode to the reconstruction of the original data can be analyzed by examining σ_i^2, where i is the i'th mode. It can be loosely called "energy" of the mode. (In the case where u is velocity, this is true.) As shown in Figure 5, the contribution of the higher modes decay rapidly. (Note the logarithmic scale.) Finally, the Galerkin simulation using the first 7 POD modes is shown in Figure 6. To recall, it is the simulation of Eq. 12, where a set of 7 ordinary differential equations is solved. As evidenced by the closeness of the Galerkin simulation (in black) and full Spectral element simulation (blue), the two simulations agree closely. The difference is on the order of 1%. In effect, the number of computational elements has been reduced from 97 (Spectral element method) to 7 (POD/Galerkin method).

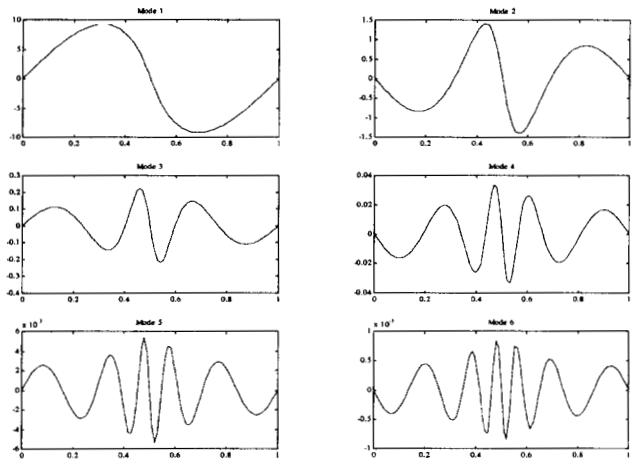

Figure 4. First six extracted modes of Burgers' Equation.

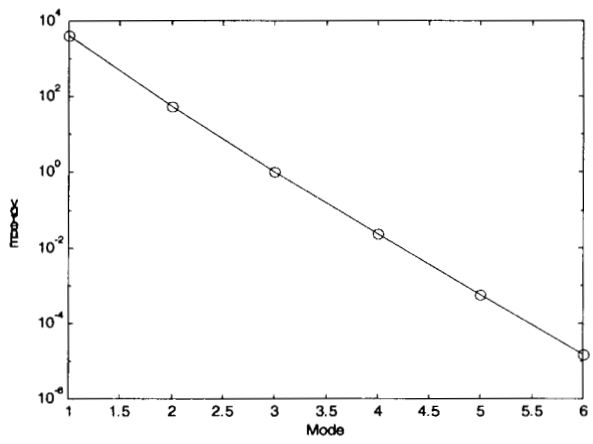

Figure 5. The "energies" of the first 6 modes of Burgers' Equation.

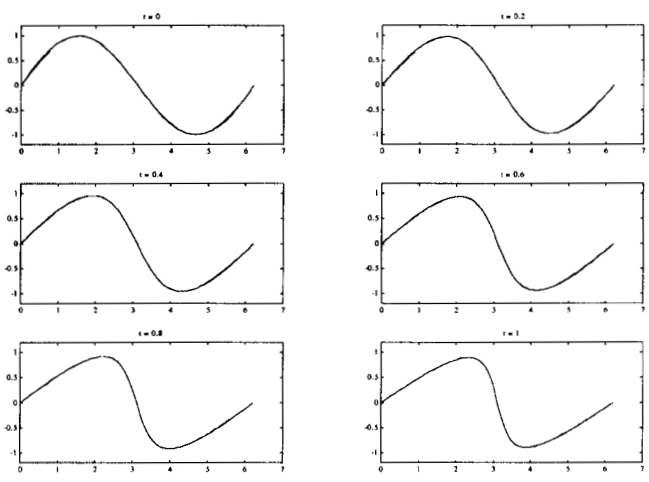

Figure 6. Galerkin simulation of Burgers' Equation using first 7 POD modes. The Galerkin simulation is shown in black. The Fourier simulation is shown in blue.

To test the robustness of the POD/Galerkin method, 10% uncorrelated Gaussian noise was added to the simulation. The extracted POD modes are shown in Figure 7. The first two modes are similar to those without noise. Only slight ripples are observed. However, the higher modes are unrecognizable. The random white noise manifests itself as higher modes. As shown in Figure 8, the contribution from the higher modes does not decay as in the case without noise. This is because white noise contributes to every frequency and POD mode.

The Galerkin simulation using the 7 noisy POD modes are shown in Figure 9. Even with the noisy modes, the Galerkin simulation (black) closely matches those of full-blown Spectral element simulation (blue). The difference is in the order of 5%. This result demonstrates that the POD/Galerkin method is robust to noise, and it can be used in conjunction with empirical data which is typically characterized by noise.

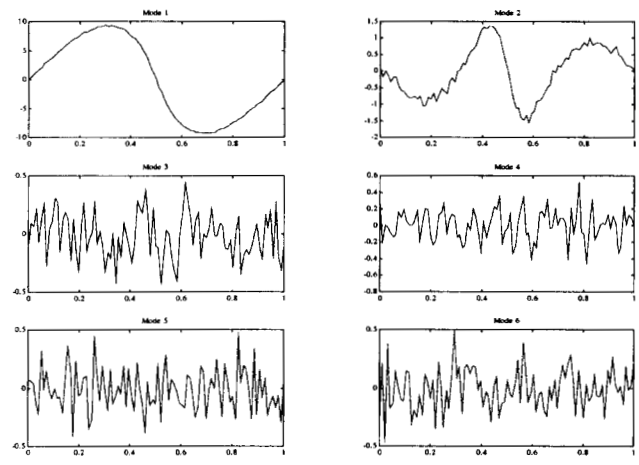

Figure 7. First six extracted modes of Burgers' Equation with 10% uncorrelated Gaussian noise.

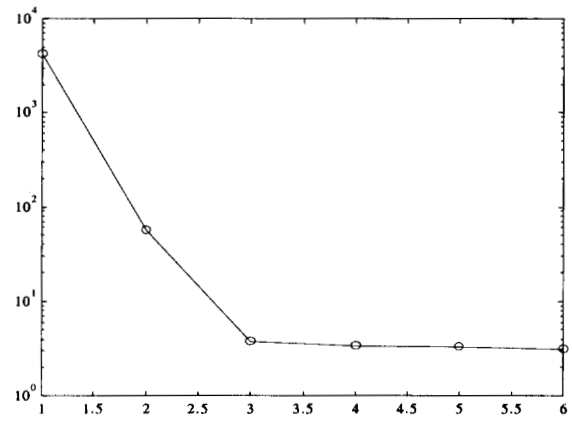

Figure 8. The "energies" of the first 6 modes of Burgers' Equation with 10% uncorrelated Gaussian noise.

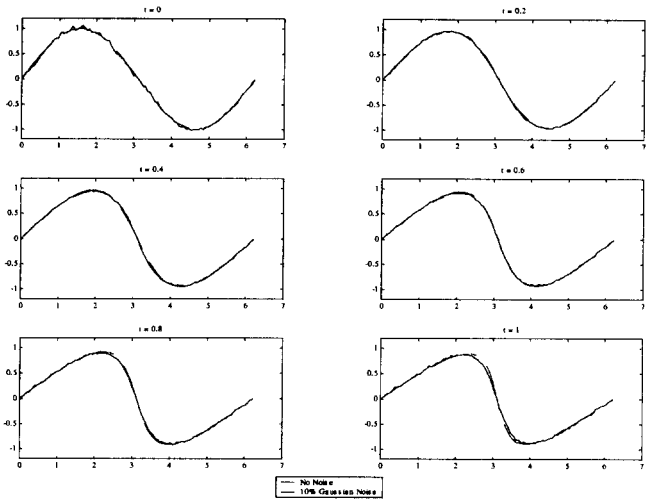

Figure 9. Galerkin simulation of Burger's Equation with 10% uncorrelated Gaussian noise using first 7 POD modes. The Galerkin simulation is shown in black. The Fourier simulation is shown in blue.

8. CONCLUSION

The procedure for creating a low-order dynamical model for a complex system using the POD method was illustrated. As an example, the POD procedure was shown on the Burgers' Equation. It was demonstrated that the POD method could be used to construct a high fidelity low-order dynamical model even in the presence of noise in the data. In the absence of noise, the lower-order model (7 POD modes) could simulate the system with 1% accuracy. Under noisy conditions (10% uncorrelated Gaussian noise), the model could simulate the system with 5% accuracy.

The POD method will serve as an important technique for the development of the Model Filter component of BEAM. In order to fully implement the Model Filter, i.e., "gray-box," fault detector, a deterministic model is necessarily. In the case where a detailed model of the system does not exist, or the model may be too complex to implement for real-time, a reliable method for generating a simplified model that accurately describes the system is required. In such case, the POD is an ideal method for creating simplified models for dynamical systems. It offers the advantage of reducing complex system models to a simple low-order dynamic model for real-time applications as well as generating models from empirical data when the physics of the system is known only partially.

ACKNOWLEDGEMENT

The authors would like to thank the generous assistance of Dr. Xia Ma at Brown University in setting up the Burgers' Equation and Galerkin simulations.

REFERENCES

[1] Mackey, R., James, M., Park, H. G., Zak, M., "BEAM: Technology for Autonomous Self-Analysis," IEEE Aerospace Conference, Big Sky, Montana, March 2001.

[2] Zak, M., Park, H. G., "Gray-box Approach to Fault Diagnosis of Dynamical Systems," IEEE Aerospace Conference, Big Sky, Montana, March 2001.

[3] Park, H. G., Mackey, R., James, M., Zak, M., Kynard, M., Greene, W., Sebghati, J., "Analysis of Space Shuttle Main Engine Data Using Beacon-based Exception Analysis for Multi-Missions," IEEE Aerospace Conference, Big Sky, Montana, March 2002.

[4] Preisendorfer, R. W., Mobley, C. D. (editor), Principal Component Analysis in Meteorology and Oceanography, New York, Elsevier, 1988.

[5] Holmes, P., Lumley, J. L., Berkooz, G., Turbulence, Coherent Structures, Dynamical Systems, and Symmetry, New York, Cambridge University Press, 1996.

[6] Whitham, G. B., Linear and Nonlinear Waves, New York, Wiley, 1974.

[7] Ma, X., Karniadakis, G. E., Park, H., Gharib, M., "DPIV/T-driven convective heat transfer simulation," *Int. J. of Heat and Mass Transfer* Vol. 45, pp. 3517-3527, 2002.

Han Park *is a senior member of the technical staff in the Ultracomputing Technologies Research Group in the Information and Computing Technologies Research Section at the Jet Propulsion Laboratory, California Institute of Technology. He joined JPL in 1999 and performs research and development of fault detection/diagnosis algorithms for aircraft and spacecraft systems. He received his B.S. in M.E. at the University of California at Berkeley, S.M. in M.E. at MIT, and Ph.D. in Aeronautics at the California Institute of Technology. His research interests are in the areas of vehicle health monitoring, signal processing, image processing, pattern recognition, color recognition, quantitative visualization, fluid mechanics, and heat transfer.*

Michail Zak *is a senior research scientist in the Ultracomputing Technologies Research Group at the Jet Propulsion Laboratory, California Institute of Technology. He has been with JPL since 1977. His research interests include non-linear dynamical system theory, chaos theory, quantum information processing, neural networks, and complex systems theory. He is the author of over 150 scientific and technical papers, and research monographs.*