SANDIA REPORT

SAND2011-0951
Unlimited Release
Printed February 2011

Probabilistic Models for Feedback Systems

Paul T. Boggs and Matthew D. Grace

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865) 576-8401
Facsimile: (865) 576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.osti.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800) 553-6847
Facsimile: (703) 605-6900
E-Mail: orders@ntis.fedworld.gov
Online ordering: http://www.ntis.gov/help/ordermethods.asp?loc=7-4-0#online

2011-0951
Unlimited Release
Printed February 2011

Probabilistic Models for Feedback Systems

Paul T. Boggs
Quantitative Modeling & Analysis (08954)
Matthew D. Grace
Scalable & Secure Systems Research (08961)
Sandia National Laboratories
Livermore, CA 94550, U.S.A.
ptboggs@sandia.gov & mgrace@sandia.gov

Abstract

In previous work, the we developed a Bayesian-based methodology to analyze the reliability of hierarchical systems. The output of the procedure is a statistical distribution of the reliability, thus allowing many questions to be answered. The principal advantage of the approach is that along with an estimate of the reliability, we also can provide statements of confidence in the results. The model is quite general in that it allows general representations of all of the distributions involved, it incorporates prior knowledge into the models, it allows errors in the "engineered" nodes of a system to be determined by the data, and leads to the ability to determine optimal testing strategies. In this report, we provide the preliminary steps necessary to extend this approach to systems with feedback. Feedback is an essential component of "complexity" and provides interesting challenges in modeling the time-dependent action of a feedback loop. We provide a mechanism for doing this and analyze a simple case. We then consider some extensions to more interesting examples with local control affecting the entire system. Finally, a discussion of the status of the research is also included.

Contents

Figures

1 Introduction

In previous work, the authors and colleagues developed Bayesian techniques for estimating the reliability of engineered systems [2]. We used a network model for the system where the leaf nodes are the inputs to the system and the internal, or "system" nodes are assumed to be designed, i.e., we know what they are supposed to do. This is not the most general model, but adequate for many interesting systems. The idea is to test the composite system at various points and estimate the reliability of the system based on these tests. We accomplish this by using polynomial chaos expansions (PCEs) [3, 6] for the random variables (RV) at the leaf nodes, Bayes' theorem [1] to determine the distribution of the PCE coefficients, given the test data, and Markov-chain Monte Carlo (MCMC) [5, 4] to sample this distribution.

In our models, reliability is defined as the percentage of time the system yields output in its acceptable range. Thus for each MCMC sample, we can produce an estimate of the reliability, which yields a distribution for the reliability.

There are many advantages of this approach for testing systems. In particular, by using PCEs, we avoid the problem of assuming a particular distribution for the random variables. This is specifically important in some situations where early attempts to gauge the reliability assumed Gaussian (normal) distributions that did not give nearly enough weight to the tails and thus overestimated the reliability. Another advantage is that our approach allows a general nonlinear function at each system node. Thus, we do not have to rely on linear models. We also note that our procedure produces a distribution of the reliability and thus allows us to provide more insight into the reliability than just a single estimate of what it is. For example, we can provide some estimate of our confidence in the estimate. Finally, we can provide a fictitious node for each system node to represent the error in that node. We assume that we cannot directly observe these nodes, but we can infer the parameters by collecting data.

We also have extended the approach to time-dependent systems. Here we provide a model of how the system varies over time and use test data to infer the parameters of this model. We show that such systems can be represented in a way that permits us to use almost all of the above analysis for estimating reliability. At the conclusion, however, we can use our model to predict the reliability at some future time. This gives a way to deal with aging issues that are extremely important in some contexts.

Finally, we can use the same framework to formulate and answer questions about an optimal testing strategy [2]. In particular, we can ask for the best place to do the next test, but here we must be careful to formulate the notion of "best" in an appropriate way. Roughly speaking, our confidence in our estimate of reliability is related to the variance of the distribution of the reliability. Thus a way to improve our confidence would be to reduce the variance, so one formulation of "best" is to determine the test that reduces the variance more than any other. Many variations on this can be included, for example, given a budget for tests, what is the best way to spend this?

An additional feature of this approach is the ability to extend it in several interesting ways. Of particular interest in the area of complexity is to extend it to systems that contain feedback. We believe that feedback is a necessary component of complexity, i.e., systems exhibiting complexity must contain feedback. (Of course, feedback is not sufficient for complexity.) In this paper, we take a first look at systems with feedback in the above framework. In section 2 we give a specification of a simple feedback model and introduce our notation. As we shall see, this requires us to be specific about timing of events in a feedback loop. To address this, we introduce a model that allows us to specify a general mechanism for turning a static feedback system into a system that is feed-forward-in-time. This allows us to use much of the same framework as described above, with the same advantages, to feedback systems. In section 4 we construct a very simple system with the property that for a certain range of input, the system is stable, but outside of that range, it diverges. Thus, in addition to predicting its reliability, we can try to predict the range of inputs for which the system is stable. This is a first attempt to try to deal with the type of behavior that can arise in complex systems; we show numerically that this is achievable. In section 5 we consider a more interesting set of functions in the feedback loop and look to see if we can obtain some unexpected behavior. (This work is not finished, yet.) Finally, in section 6 we discuss the current status of the work and future directions.

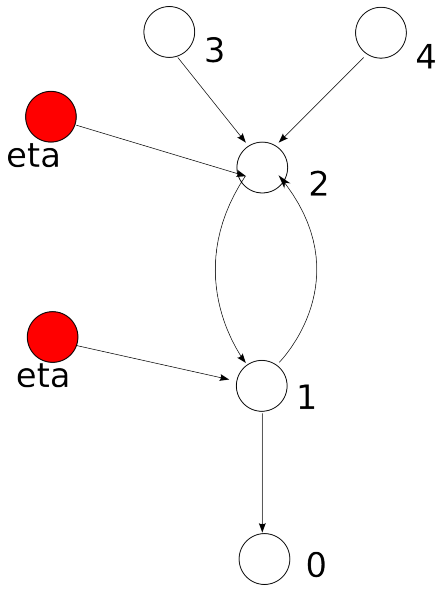


Figure 1: System with feedback. The nodes labeled η are the error nodes.

2 Model and Notation

The task is to create a model for feedback systems. The initial models should be simple enough so that we can develop appropriate algorithms and do some detailed testing. Figure 1 contains a simple feedback loop between nodes 1 and 2. To be specific, we refer to nodes 3, 4, and both η-nodes as leaf nodes. Nodes 1 and 2 are system nodes, and node 0 is the output node. The idea is that nodes 3 and 4 provide input to node 2. Node 2 then begins the feedback with node 1. The η nodes represent the errors that are produced by nodes 1 and 2. This way of modeling allows us to have a fairly general way to incorporate errors into the process. Note that the η-nodes cannot be observed directly, but only inferred from the data.

Let the output of node i be given by Ω_i. At leaf node i, the output is a RV represented by a PCE. In particular,

$$\Omega_i = \sum_{j=0}^{\infty} \theta_{ij} H_j(\xi_i), \tag{2.1}$$

where the θ_{ij} are real numbers, H_j is the jth Hermite polynomial, and $\xi_i \sim N(0,1)$, i.e., a normally distributed RV with mean 0 and variance 1. Any RV with finite variance can be represented arbitrarily accurately (in distribution) with sufficiently many terms of this series. An advantage of this formulation is that it already contains a distinction between epistemic and aleatory uncertainty: The values of the coefficients θ_{ij} are improved with increasing data, but the overall variation in Ω_i, even with perfect knowledge of the θ_{ij}, cannot be reduced.

The output at system node i is an arbitrary function of its input (or parent) nodes. Here, we have to be careful in specifying this function in order to be clear about the way in which the feedback system operates. We specify a general framework that can be specialized as

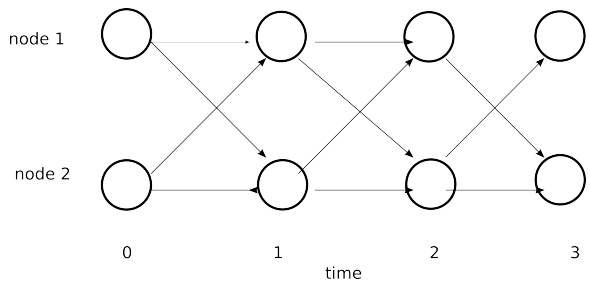


Figure 2: The time dependence diagram depicting the flow of information over time.

appropriate.

Assume that each node outputs a value at each clock cycle. We are primarily concerned about the feedback loop consisting of nodes 1 and 2; we show a schematic of the loop over time in figure 2. We define the action of node 2 at each time step as follows:

$$\Omega_2^{t+1} = \begin{cases} \Omega_2(\Omega_3^t, \Omega_4^t, \eta_2^t, t) & \text{if } \Omega_1^t = \oslash \\ \Omega_2(\Omega_1^t, \Omega_3^t, \Omega_4^t, \eta_2^t, t) & \text{else} \end{cases} . \tag{2.2}$$

Similarly, we can define the action of node 1:

$$\Omega_1^{t+1} = \begin{cases} \oslash & \text{if } \Omega_2^t = \oslash \\ \Omega_1(\Omega_2^t, \eta_1^t, t) & \text{else} \end{cases} . \tag{2.3}$$

Several comments on this loop are in order:

- We assume that at $t = 0$ node 2 has values of Ω_3^0, Ω_4^0, and η_2^0, but that $\Omega_1^0 = \oslash$. Node 2 thus evaluates $\Omega_2^1 = \Omega_2(\Omega_3^0, \Omega_4^0, \eta_2^0, 0)$; we think of this as the initialization step. This is completed at the end of time step 0.

- Node 1 at $t = 0$ has $\Omega_2^0 = \oslash$ and thus evaluates $\Omega_1^1 = \oslash$.

- At time $t = 1$, node 2 calculates $\Omega_2^2 = \Omega_2(\Omega_3^1, \Omega_4^1, \eta_2^1, 1)$ since $\Omega_1^1 = 0$ and node 1 evaluates $\Omega_1^2 = \Omega_1(\Omega_2^1, \eta_1^1, 1)$.

- By construction here, we note that at every other time step, the evaluations at both nodes 1 and 2 are not necessary. Some sort of a "wait" state could be used or some other calculation could be done.

- We have explicitly included a dependency on time in the evaluations at nodes 1 and 2. This, of course, allows a general time dependency in the system, but could be ignored to study a steady-state system, i.e., a system in which aging effects or other time-dependency takes place on a scale much longer than the time scale of the loop.

- We have not specified how this loop is terminated and output of the system is finally achieved. We address this point next.

To get output from node 0, we need to specify when output from node 1 is available to node 0. For example, we could postulate the existence of a general conditional, say $\mathscr{C}$, which, when satisfied, implies that node one makes its state available to node 0. Consider this extension to the evaluation at node 1:

$$\Omega_1^{t+1} = \begin{cases} \oslash & \text{if } \Omega_2^t = \oslash \text{ and } \mathscr{C} \text{ is not satisfied} \\ \Omega_1(\Omega_2^t, \eta_1^t, t)) & \text{if } \mathscr{C} \text{ is not satisfied} \\ \Omega_1^F(\Omega_2^t, \eta_1^t) & \text{if } \mathscr{C} \text{ is satisfied} \end{cases} . \tag{2.4}$$

This is coupled with the specification that node 1 controls its output so that Ω_1^t goes to node 2 if $\mathscr{C}$ is not satisfied and to node 0 if it is.

Remarks:

- There is no restriction that the output of any of the nodes is a scalar value — it could be a vector.
- The conditional $\mathscr{C}$ could contain several possibilities, e.g., output must happen every τ time steps, or output happens when the loop "converges" (however that is specified), or when the phase of the moon is appropriate.
- We can consider situations where the whole process is driven by white noise represented by the ξ RVs in the leaf nodes. This corresponds to some models in the literature.

3 Research Questions

- Suppose that the iterative procedure is "stable" for $(\Omega_3, \Omega_4) \in \mathscr{S}$. Here we need to be specific about what we mean by stability. It could mean, for example, that the loop outputs a value that is in an acceptable range. Can we determine $\mathscr{S}$ in the presence of the errors represented by the η nodes? Can we tell if we are close to the boundary of $\mathscr{S}$?
- Can we determine if the system nodes are, by themselves, reliable? That is, can we estimate the associated η node values?
- How does the set of conditions, $\mathscr{C}$ affect the performance of the system?
- What is a reasonable set of functions on which to begin the analysis?
- In what ways should this be extended to create increasingly complex systems.

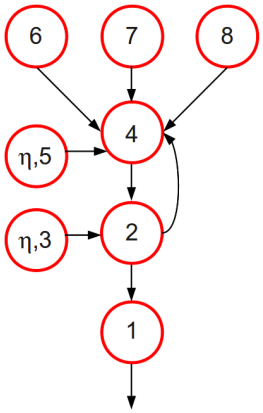


Figure 3: A simple feedback loop with errors. The η-nodes are specified.

4 First Example

4.1 Simple feedback system with errors

We first specify a very simple fixed-point iteration that can be used in a feedback loop. The guiding idea is that the feedback loop is designed to stabilize a process. A target T is provided and the output of the feedback loop is supposed to move its output closer to the target. With sufficient time, it could, in theory, get arbitrarily close if the process is convergent.

Consider the iterative procedure

$$s_{k+1} = s_k - 2\lambda(s_k - T), \tag{4.1}$$

where the s_k are the outputs of the system. Assume that the initial state, s_0 has been provided. The value λ is the control parameter with the properties that Eq. (4.1) is convergent if $\lambda \in (0, 1)$ with optimal value at 0.5. Note that this convergence result follows from the fact that

$$\frac{(s_{k+1} - T)}{(s_k - T)} = 1 - 2\lambda. \tag{4.2}$$

The idea here is that our controller will provide the step λ to the node that will compute the next value s_k, but that it will operate with error. Suppose we have a bound: $\overline{\lambda}$ such that all steps $\leq \overline{\lambda}$. The input to the controller is the target T, $\overline{\lambda}$, and starting value s_0, all with error, i.e., all are RVs.

We will implement the feedback loop as follows (see figure 3). Node 4 is the controller and node 2 computes the next iterate. In particular node 2 has input s and λ and computes s^+ which it returns to node 4 with error from node 3.

Node 4 does the following: At the first iteration, it takes in T, s_0, and $\overline{\lambda}$ (from nodes

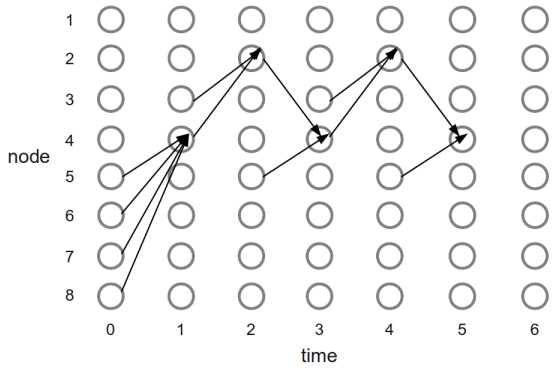


Figure 4: Time-dependence diagram for simple feedback system. This process can be continued arbitrarily to the right.

6, 7, and 8, respectively. It then sets $\lambda = \overline{\lambda}$, adds an error, adds an error to s_0 and sends this to node 2. When it gets the s_+ (with error) back from node 2, it evaluates the above ratio (4.2), and adjusts λ. The current implementation of this is given in feedback.m. The implementation at node 2 is given in iter.m, and the input from nodes 6, 7, and 8 is given in feedTest.m.

Currently, this uses normally distributed errors for everything, but can be modified to obtain the performance one desires. Finally, we limit the number of times the feedback loop is executed to K iterations. We note, of course, that this makes the system bounded, but we could impose something like a convergence criterion that would yield an unbounded loop, but for this simple example, this would not change much.

4.2 Computational Procedure

Here we specify more of the details of the procedure. We first specify the "true" system by specifying an order and appropriate coefficients for the PCE at each leaf node. The system nodes are as defined above. Given the "true" system, we can compute a sample at each leaf node and propagate this through the system. The best model for following the computations is the time-dependence diagram in figure 4. After K passes through the feedback loop, node 1 outputs a result which is the ratio

$$r_i = \frac{|s_{k+K} - T|}{|s_k - T|} \tag{4.3}$$

where i is the index of the sample. Since the procedure is converging if $r_i < 1$, we can compute the reliability of the system computing as

$$r = \frac{\text{no. of } r_i < 1}{\text{total } r_i}. \tag{4.4}$$

At this point we can specify a set of data that is collected. This will include the nodes, time, and number of replications. We will then use this data to determine the properties

of the system. In particular, we propose a Bayesian approach to predict the system. The details of this approach are in [2]. An outline is as follows:

- Specify an initial guess at the order and coefficients of the PCEs for each node.
- Specify a prior on the coefficients.
- Use Bayes Theorem to get an expression for the joint probability of the coefficients, given the data, i.e.,
$$\pi(\theta|d) \propto \pi(d|\theta)\pi(\theta), \tag{4.5}$$
with the usual interpretation of these quantities.
- Use Markov chain Monte Carlo (MCMC) to sample this distribution
  - For each sample θ, do the above procedure to compute an estimate of the reliability
  - At the end, plot the distribution of the reliability given the data. We can also give a distribution of the coefficients θ.

Another thing we want to be able to do is to estimate the value of the bound, $\overline{\lambda}$, which separates stable behavior ($r_i < 1$) from the unstable behavior. Without error, this is trivial, but with error we will not see a definite break, but a region about 1 where things are not certain. To get an estimate of the break-point, call it λ^*, we propose plotting the r_i as a function of the bound $\overline{\lambda}$ and using least squares to estimate λ^*. We can also estimate the optimal step ($\lambda_{\text{opt}} = 0.5$) by the same procedure. Thus, we modify the last step above to compute these quantities and thereby obtain distributions for λ^* and λ_{opt}.

4.3 Results

A code was written in Matlab to implement the procedure described above. The goal here was to simply test the algorithm and to see if the procedures for estimating the cross-over point from stability to instability. Thus, no attempt was made to maximize efficiency. (We address this issue in section 6.)

We used just two terms in the PCEs for the leaf nodes and started very close to the true solution. Tests were made at nodes 2 and 3 at all iterations of the feedback loop; 50 tests were made at each. We also did 50 replications of tests at nodes 5-7. We limited the number of feedback iterations to 3. With 100 iterations of the MCMC code, we obtained reasonable results. Clearly we need much more testing to make stronger claims about the efficacy of the ideas. The only claim made here is that the procedure seems to work; it produces reasonable results as seen in figures 5 – 7. Figure 5 shows the reliability distribution centered around .8, which was the chosen reliability for the system. Figure 6 shows the distribution of the computed cross-over between stability and instability. As mentioned above, this value should be 1; the distribution is clearly centered there. Finally, figure 7 shows the distribution of the estimate of the optimal value. This overestimates the correct value of .5, but the fact that

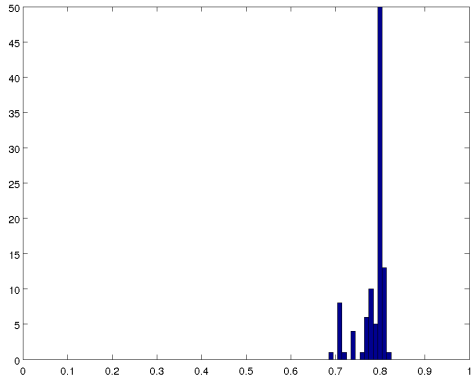


Figure 5: Reliability Distribution

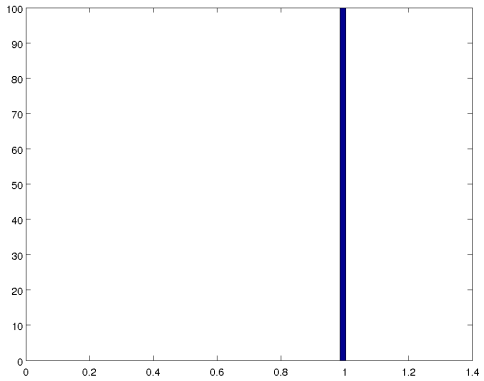


Figure 6: Stability cross-over

the accuracy here is somewhat poor is due to the fact that almost no input data was in this range. In summary, this simple example illustrates that the procedure can be effective.

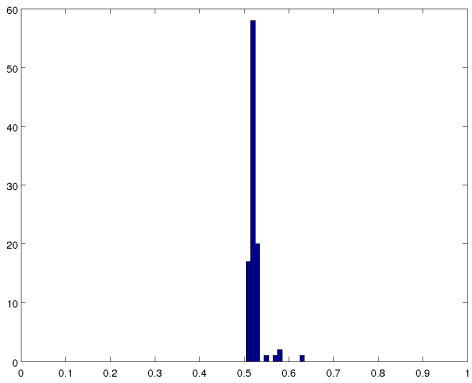


Figure 7: Optimal Input

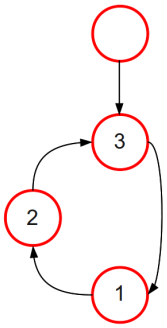


Figure 8: Three-Node feedback with logistics maps.

5 Example with Logistics Map

In this section, we describe some experiments with a more complicated and interesting feedback system based on the well known logistic maps. For a brief introduction, see

`http://en.wikipedia.org/wiki/Logistic_map.`

Briefly, the logistic map represents a dynamical system described by a polynomial of degree 2. The interesting feature is the level of complexity that can be seen by varying a parameter that occurs in the formula.

To be specific, the logistic map is taken to be the nonlinear recurrence relation

$$x_{k+1} = r x_k (1 - x_k), \tag{5.1}$$

where r is the parameter that controls the process. It is easy to verify that for $r \in (0, 4)$ the process remains bounded. For $r < 3$ the process will converge to a single value; for r between 3 and 3.45 (approximately), the process will oscillate between two values; for r between 3.45 and 3.54 (approximately) the process will oscillate between four values. As r increases points will be reached where the process will oscillate between eight, then 16, then 32, etc. values until when r is larger than 3.57 almost all initial values result in no finite number of oscillations. There are, however, certain "islands of stability" for large values of r where oscillations among 3, or some other number of values is possible.

We create a feedback loop with logistic maps at each node. For example, consider the 3-node feedback loop in figure 8. A value of r is set for each node. Assume that the process is started at node 1. Node 1 does one iteration and sends its output to node 2, which does one iteration and sends its output to node 3, etc. We did some experimentation with different values of r at each node and obtained some results that are not obvious. For example, with $r_1 = 3.4$, $r_2 = 3.6$, and $r_3 = 3.8$ we obtained the plot in figure 9 that shows a stable oscillation among 3 values. Note that this is not possible for any of the three values of r individually. (To obtain this plot, we sorted the values and plotted them.)

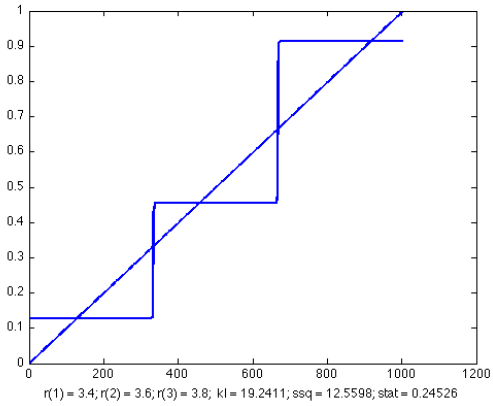


Figure 9: Three-node feedback using logistic maps with $r_1 = 3.4$, $r_2 = 3.6$, and $r_3 = 3.8$

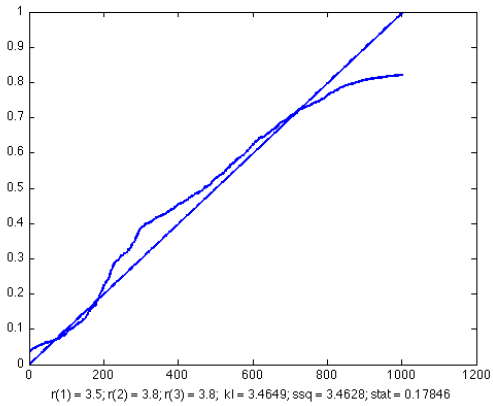
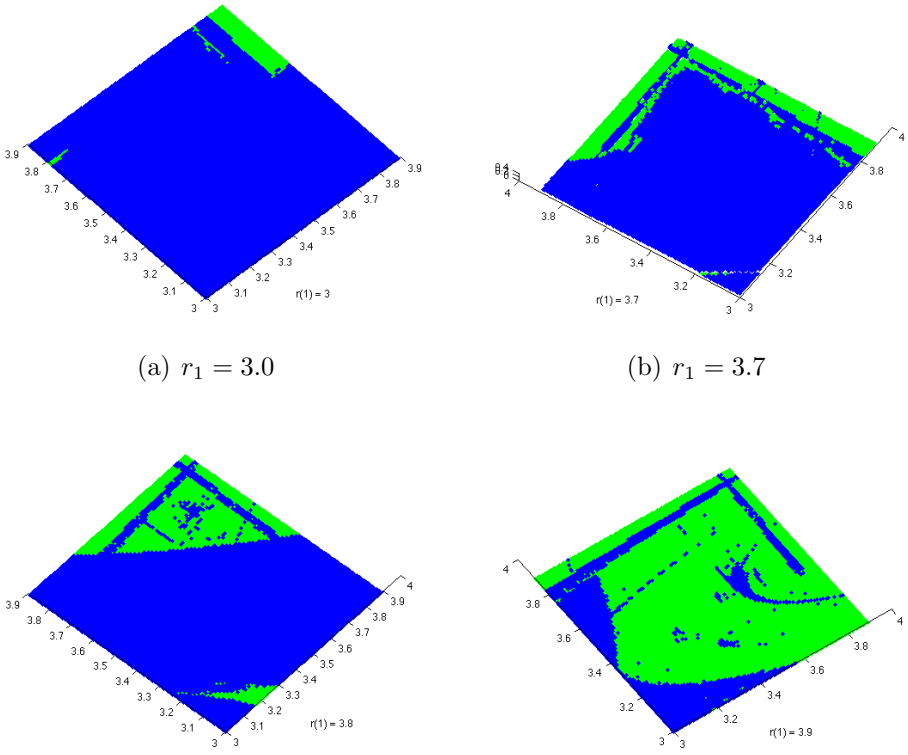


Figure 10: Three-node feedback using logistic maps with $r_1 = 3.5$, $r_2 = 3.8$, and $r_3 = 3.8$

In the course of these experiments, we observed many triples of the r_i values that appeared to be generating a chaotic plots, i.e., ones with no stable oscillations (or perhaps too many to see). Thus we wanted to create a test to measure the "distance" from such plots to plots exhibiting chaos. Note that if we sorted the values from a chaotic result and plotted them, we would obtain (approximately) the straight line from $(0, 0)$ to $(1, 1)$. If we consider our plots to be cumulative distribution functions, we can use the Kolmogorov-Smirnov (KS) test to measure the distance from these distributions to a true uniform distribution. In order to make these more meaningful, we had to shift the data to properly overlap the uniform data. The straight line in figure 9 is the uniform line. The value the KS statistic is .245. By contrast, the plot for $r_1 = 3.5$, $r_2 = 3.8$, and $r_3 = 3.8$ if given in figure 10 and the KS stat is .178.

We investigated two items of interest. First, we wanted to know if we could neatly partition the 3-dimensional space consisting of (r_1, r_2, r_3) into two subsets where the KS statistic was above (below) a certain value. Thus we made a series of plots where r_1 was fixed and r_2 and r_3 were varied between 3.0 and 3.9 by .01. For each point, we placed

a blue dot if the KS statistic was $> .2$ and a green dot if not. It was not obvious how this would turn out. Four interesting plots are given in figure 11. These plots correspond to $r_1 = 3.0,\ 3.7,\ 3.8$, and 3.9. Note that even for $r_1 + 3.0$, where we would expect more stability, there are some interesting islands and strips of green into predominately blue regions. For higher values of r_1 we obtain more interesting intermixing of the green and blue, suggesting that, by this measure, there are regions where it would be very hard to predict what would happen. We speculate that such complexity would be more pronounced as the model is scaled up to include more nodes.



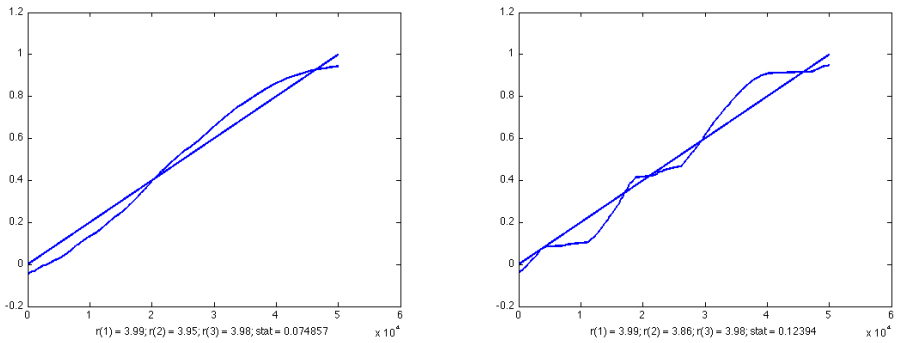
(a) $r_1 = 3.0$ (b) $r_1 = 3.7$

(c) $r_1 = 3.8$ (d) $r_1 = 3.9$

Figure 11: Plots showing regions where the KS statistic is $> .2$ in blue and $\leq .2$ in green.

The next item of interest is to see the effect of trying to control one node in the system. That is, we control one node based only on information at that node, with no regard to what the other nodes are doing. Of course, the input to the node to be controlled is dependent on the other nodes. To be specific, we put our control at node 2. Our first attempt at this was to collect some number of inputs to node 2, calculate the KS statistic and try to adjust r_2 to keep the KS statistic within a specified range. We did a number of tests and settled on a range .35 to .40. If the statistic was below .35, we decreased r_2 by .01; if it was greater than .40, we increased it by .01. In figure 12 we show a run with no control and then a run with the control activated. We see that for the initial values of the r values shown, with no control we obtain a value of the KS statistic of .07 but with control at only node 2 we obtain a value of .12. There appears to be much more structure in the controlled plot. This shows

that we can have a profound influence on the whole system by controlling just one node. In some sense, this is akin to a "canalizing" parameter in Boolean networks.



(a) Uncontrolled System (b) Controlled System

Figure 12: The stability of the system based on controlling node 2 only.

These results suggest some additional areas of research. As suggested earlier in this paper, we could add errors in various places and try to determine the effect on the system. We could make the range variables part of the input and try to determine ranges for which control can be achieved. We could allow "drift" in the r values and try to adapt the controls to deal with this. Finally, there are many questions about how this could be scaled. For example, how many nodes could be controlled by just one node?

6 Discussion

This is a preliminary report on our attempts to extend our tools for handling epistemic uncertainty in hierarchical systems to systems that contain feedback. We showed how to model feedback in such systems by a feed-forward in time model that allows us to use many of the tools developed in previous work. We then considered feedback models based on the well known logistic map that demonstrates a transition to chaos as its controlling parameter is increased. By coupling such maps in a feedback loop, we can obtain unexpected behavior in the final output. By adding an additional feedback control on one node, we showed that we can control the overall output of the system. We speculate that by scaling up such a system, we will be able to create unexpected behavior and it will be interesting to see what percentage of nodes must be controlled to control the behavior of the whole system.

Finally, the amount of computation rapidly increases as we increase the number of nodes and the length of the feedback cycle. Thus it is of interest to examine carefully the algorithmic aspects of our procedure. We note here that there are many possibilities for parallelism, especially in the computation of the likelihood function, which is the most expensive part of the calculation. In fact, these calculations are "embarrassingly" parallel, so much could be done to speed the calculations and thus extend the range of our techniques.

References

[1] James O. Berger. *Statistical decision theory and Bayesian analysis*. Springer series in statistics. Springer, New York, NY, 2nd edition, 1985.

[2] Paul T. Boggs, Matthew D. Grace, Youssef M. Marzouk, Philippe P. Pébay, John Red-Horse, James T. Ringland, Kathleen Diegert, and Rena Zurn. Practical reliability and uncertainty quantification in complex systems: Final report. Technical Report SAND2009-6072, Sandia National Laboratories, 2009.

[3] Thierry Crestaux, Olivier Le Maître, and Jean-Marc Martinez. Polynomial chaos expansion for sensitivity analysis. *Rel. Eng. & Sys. Safety*, 94(7):1161–1172, 2009.

[4] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov chain Monte Carlo in practice*. Interdisciplinary Statistics. Chapman & Hall/CRC Press, New York, NY, 1996.

[5] F. James. Monte Carlo theory and practice. *Rep. Prog. Phys.*, 43(9):1145–1189, 1980.

[6] Youssef M. Marzouk and Habib N. Najm. Dimensionality reduction and polynomial chaos acceleration of bayesian inference in inverse problems. *J. Comp. Phys.*, 228(6):1862–1902, 2009.

Sandia National Laboratories