LA-UR-09-***** 00419

Approved for public release; distribution is unlimited.

Title:

Illustrating the Future Prediction of Performance Based on Computer Code, Physical Experiments, and Critical Performance Parameter Samples

Author(s):

Michael S. Hamada
David M. Higdon

Intended for:

Submission to Quality Engineering
January 2008

Illustrating the Future Prediction of Performance Based on Computer Code, Physical Experiments, and Critical Performance Parameter Samples

M.S. Hamada and D.M. Higdon
Statistical Sciences Group
Los Alamos National Laboratory

01.23.09 0630

Abstract

In this paper, we present a generic example to illustrate various points about making future predictions of population performance using a biased performance computer code, physical performance data, and critical performance parameter data sampled from the population at various times. We show how the actual performance data help to correct the biased computer code and the impact of uncertainty especially when the prediction is made far from where the available data are taken. We also demonstrate how a Bayesian approach allows both inferences about the unknown parameters and predictions to be made in a consistent framework.

Keywords: Bayesian, Bias correction, Markov chain Monte Carlo, Monte Carlo.

Introduction

The challenge we faced recently was to develop a generic example to explain to upper management various aspects about prediction of a population's performance in the future. It is particularly important that stakeholders, managers and decision makers understand the role of various information sources in assessing the future performance of the population of interest. With better understanding of the values of these information sources, better decisions can be made regarding the allocation of scarce resources to ensure future predictability. The prediction using Bayesian methods integrates performance computer code, physical performance data from experiments, and sample data of the computer code input from the population at a few times.

The following are various aspects of the generic example and points that we wanted to highlight through the generic example:

- A computer code models the performance of a system given inputs (critical performance parameters) that characterize a system. The code, because it is a model, is necessarily biased relative to actual performance.
- For a given set of input values, the actual performance varies and is described by a probability distribution. We want to integrate actual performance data from experiments that varied the input values. In the integration, we attempt to correct the bias of the computer code relative to actual performance.
- The input values vary across a population and is described by a probability distribution. Moreover, the population's distribution of the input values changes over time. Consequently, the population's performance distribution obtained by propagating the input distribution through the performance relationship changes over time.
- Available are input data sampled from the population at different times. We use these data to predict the input distribution as it changes over time.
- Because we use data to estimate the bias of the computer code, performance variation, and input distribution parameters as it changes over time, we want to account for the uncertainty in the estimation. In predicting into the future, we extrapolate beyond the available data and show the increased uncertainty from extrapolation.
- In using Bayesian methods for estimation and Monte Carlo for propagating input distribution through the performance relation, we want to show a consistent and proper approach of handling uncertainty as it arises in various components of prediction.

Next, we introduce the specific example and consider these aspects and points in turn.

A Generic Example

We begin by introducing the generic example. The performance y is a function of a critical performance parameter (CPP) x. The true relation between y and x is nonlinear. Moreover, for a given x, the performance y varies as described by a probability distribution. See Figure 1 for a graph of the performance distribution as a function of x. Note that the performance distribution is assumed normal with mean $\mu(x) = 4\log(x)$ and standard deviation 0.5.

Notationally,

$$y \sim Normal(4\log(x), 0.5^2). \tag{1}$$

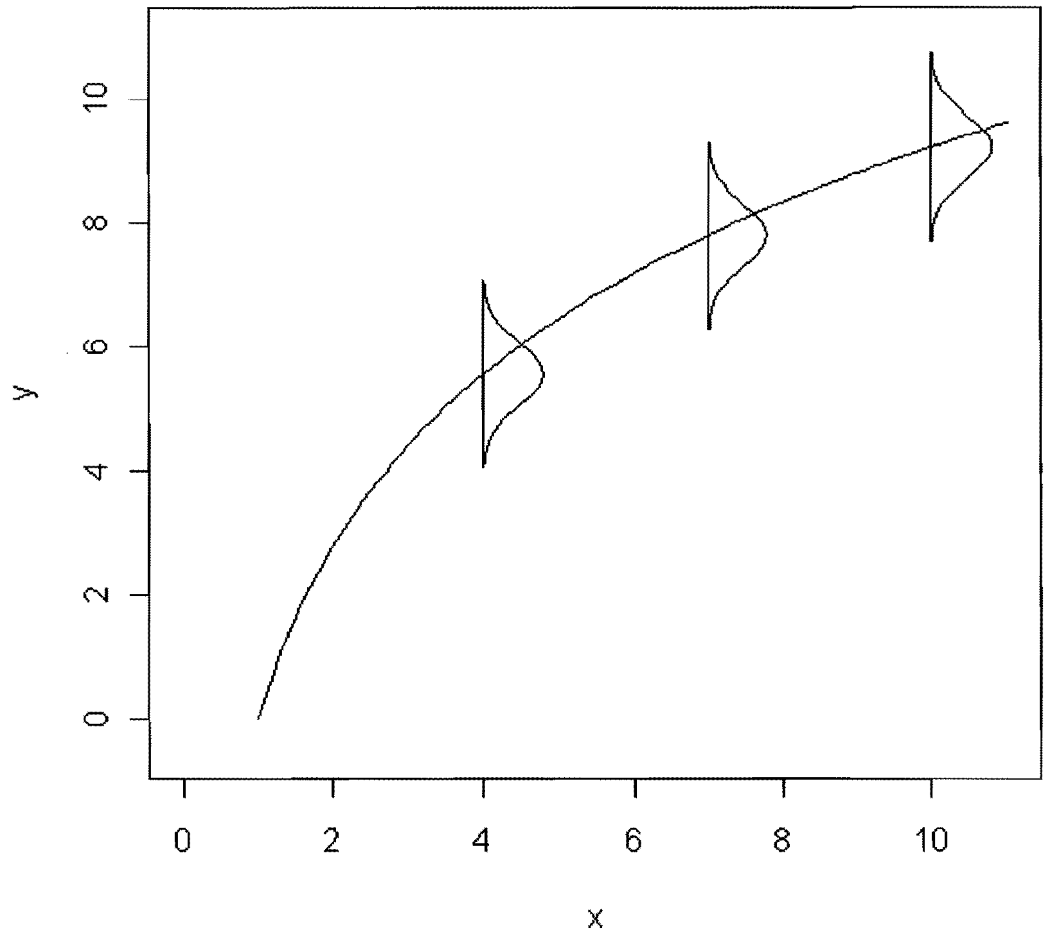

Figure 1: True performance y distribution as a function of input x.

The CPP x varies across the population and is described by a probability distribution. Moreover, the CPP distribution changes over time t. See Figure 2 for a graph of the CPP distribution as it changes over time. Note that the CPP distribution is assumed normal with mean and standard deviation 0.5/3. Notationally,

$$x(t) \sim Normal(8.5 - 0.5t, (0.5/3)^2). \tag{2}$$

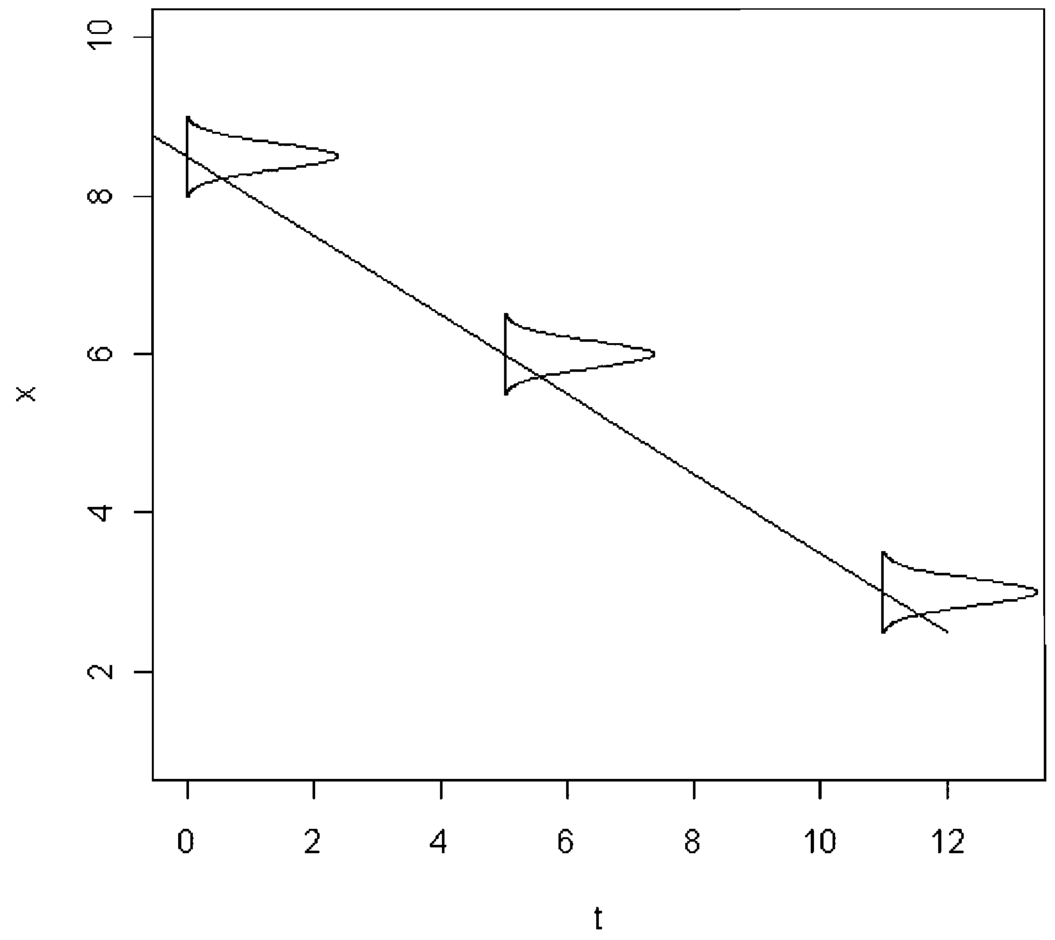

Figure 2: Critical performance parameter x distribution as a function of time t.

To calculate the performance y distribution of the population at a given time t, we use Monte Carlo as follows. Repeatedly draw an x using Equation 2 and then a y using Equation 1 with the drawn x. Do this at different times t. See Figure 3 for a graph of the performance y distribution of the population at times t equals 0 and 11. Note that the true performance distribution as a function of a given value of x appears as the nonlinear mean (solid curve) with the distribution now depicted as vertical solid lines instead of the normal distribution function as in Figure 1.

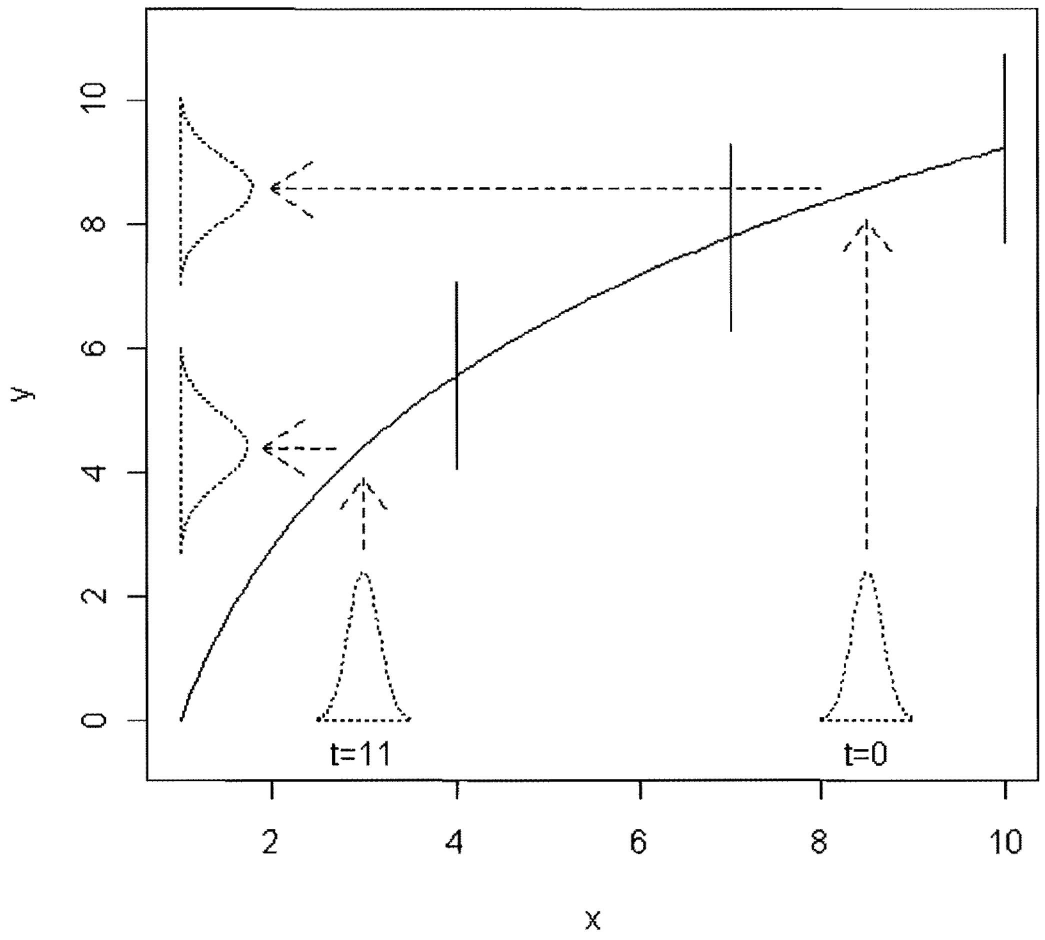

Figure 3: Performance y distribution of the population at time t equals 0 and 11 from the CPP x distribution of the population.

It is the performance distribution of the population in Figure 3 that we want to estimate based on a performance computer code and available actual performance data as well as sampled CPP values from the population at various times. Figure 4 displays the mean of the true performance distribution as a solid line. Five actual performance data are available and appear as circles around the solid line and are listed in Table 1. The computer code calculated performance ($\eta(x) = 6.0 + 0.35x$) appears as a dashed line. We assume that the computer code can easily be run at any x value. Note the bias in the computer code, especially for low x values. Finally, two CPP distributions at times 0 and 2 are depicted from right to left, respectively. Each CPP distribution is sampled five times with the sampled values appearing as circles. The sampled CPP data are listed in Table 2.

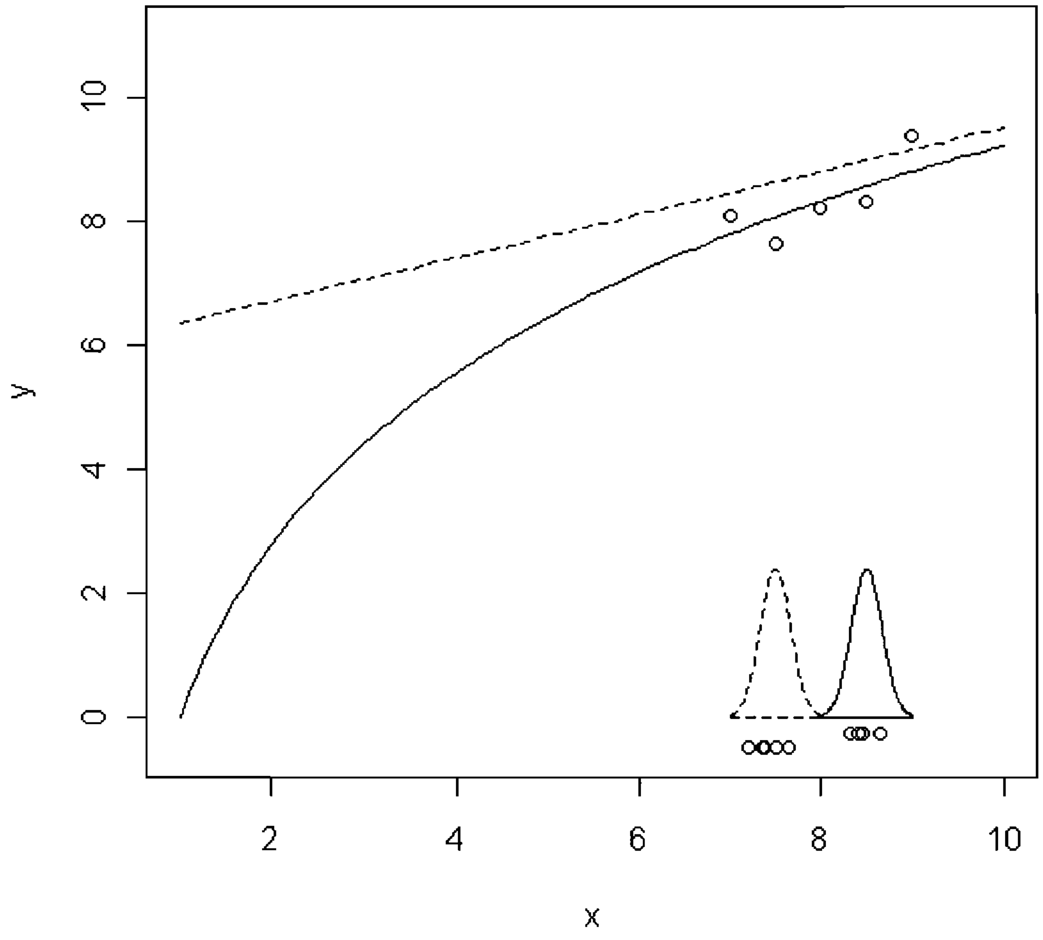

Figure 4: The mean of the true performance distribution (solid line) and five actual performance data (circles), the computer code calculated performance (dashed line), and two CPP distributions at times 0 and 2 with samples of size five each (circles).

Table 1: Actual performance y data at certain values of CPP x

x	y
7.0	7.428685
7.5	8.314868
8.0	7.865882
8.5	8.283223
9.0	9.125219

Table 2: CPP x samples at t=0 and t=2

t=0	t=2
8.418151	7.195434
8.445346	7.506274
8.640867	7.639724
8.453777	7.375526
8.323710	7.346231

Next, we consider analysis of these data as we proceed towards prediction. First, consider the actual performance data in Table 1. We assume that these data follow the model

$$y \sim Normal(\mu(x), \sigma^2), \tag{3}$$

where $\mu(x)$ is the mean of the performance distribution at x and σ^2 is the variance. From Figure 4, we observe that the computer code $\eta(x)$ as the dotted line is biased high with respect to the performance data. Consequently, we want to adjust the computer code by a discrepancy $\delta(x)$. That is, $\mu(x) \approx \eta(x) + \delta(x)$. Based on the performance data, we think the discrepancy is linear with the form $\delta(x) = \gamma_0 + \gamma_1 x$. Consequently, the assumed model for the performance data used in the analysis takes the form

$$y \sim Normal(6.0 + 0.35x + \gamma_0 + \gamma_1 x, \sigma^2). \tag{4}$$

For the sampled CPP data in Table 2, we assume the following model

$$x \mid t \sim Normal(\lambda(t), \tau^2) \tag{5}$$

with $\lambda(t) = \kappa_0 + \kappa_1 t$. That is, at a given time *t*, the CPP population distribution is normal with mean $\lambda(t)$ and variance τ^2. This is also the model that generated the sampled CPP data.

In analyzing these data with the models in Equations 4 and 5, we obtain inferences about the parameters γ_0, γ_1, and σ^2 in Equation 4 and the parameters κ_0, κ_1, and τ^2 in Equation 5. We analyze these data using a Bayesian approach, which we describe next.

Bayesian Analysis

For the performance data, let $\mathbf{\Theta}$ denote the vector of model parameters. Here, $\mathbf{\Theta}' = (\gamma_0, \gamma_0, \sigma^2)$. The Bayesian inferential approach combines prior information about $\mathbf{\Theta}$ with the information contained in the data. The prior information is described by a prior density $\pi(\mathbf{\Theta})$ and summarizes what is known about the model parameters before any data are observed. Here, we assume that little is known, and hence we choose diffuse proper prior distributions, which allow for the possibility of a wide range of values for the model parameters. The information provided by the data is captured by the data sampling model $f_y(\mathbf{y} \mid \mathbf{\Theta})$ known as the likelihood. The combined information is described by the posterior density, $\pi(\boldsymbol{\Theta} \mid \mathbf{y})$. We evaluate the posterior density using Bayes' Theorem (Degroot (1970)) as

$$\pi(\boldsymbol{\Theta} \mid y) \propto f_y(y \mid \boldsymbol{\Theta})\pi(\boldsymbol{\Theta}) \tag{6}$$

For the performance data, the likelihood $f_y(\mathbf{y} \mid \boldsymbol{\Theta})$ has the form

$$f_{\mathbf{y}}(\mathbf{y} \mid \boldsymbol{\Theta}) = \prod_{i=1}^{5} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2}[y_i - (6.0 + 0.35x_i + \gamma_0 + \gamma_1 x_i)]^2\right).$$

The prior density $\pi(\boldsymbol{\Theta})$ has the form

$$\pi(\boldsymbol{\Theta}) = f_{\gamma_0}(\gamma_0) f_{\gamma_1}(\gamma_1) f_{\sigma^2}(\sigma^2),$$

where $f_{\gamma_0}(\gamma_0)$ and $f_{\gamma_1}(\gamma_1)$ are Normal(0,1000^2) densities and $f_{\sigma^2}(\sigma^2)$ is an InverseGamma(0.001, 0.001) density; all these prior densities are proper and diffuse.

Similarly, for the CPP data, where $\boldsymbol{\Theta}' = (\kappa_0, \kappa_0, \tau^2)$, the likelihood $f_x(\boldsymbol{x} \mid \boldsymbol{\Theta})$ has the form

$$f_{\boldsymbol{x}}(\boldsymbol{x} \mid \boldsymbol{\Theta}) = \prod_{i=1}^{10} \frac{1}{\sqrt{2\pi\tau^2}} \exp\left(-\frac{1}{2\tau^2}[x_i - (\kappa_0 + \kappa_1 t_i)]^2\right),$$

where the first five data correspond to the x values at $t = 0$ and the second five data correspond to the x values at $t = 2$. The prior density $\pi(\boldsymbol{\Theta})$ has the form

$$\pi(\boldsymbol{\Theta}) = f_{\kappa_0}(\kappa_0) f_{\kappa_1}(\kappa_1) f_{\tau^2}(\tau^2),$$

where $f_{\kappa_0}(\kappa_0)$ and $f_{\kappa_1}(\kappa_1)$ are also Normal(0,1000^2) densities and $f_{\tau^2}(\tau^2)$ is also an InverseGamma(0.001, 0.001) density.

When the form of the posterior density in Equation 6 is well known, the distributional form of the posterior can be obtained in closed form. For more general forms of the posterior density, we can use recent advances in Bayesian computing to approximate the posterior distribution via Markov chain Monte Carlo (Gelfand and Smith (1990), Casella and George (1992), Chib and Greenberg (1995)). That is, Markov chain Monte Carlo (MCMC) algorithms produce samples from the joint posterior distribution of $\boldsymbol{\Theta}$ by sequentially updating each model parameter conditionally on the current values of the other model parameters. These samples of the posterior of $\boldsymbol{\Theta}$ are easy to work with making predictions, which are functions of $\boldsymbol{\Theta}$.

To analyze the performance and CPP data, we used WinBUGS (Spiegelhalter, Thomas, Best, and Lunn (2004)). See the Appendix for the two WinBUGS codes for analyzing the performance and CPP data, respectively. We used WinBUGS to generate 10,000 draws of $\boldsymbol{\Theta}$. See Figure 5 for the traces as these samples were draws and Figure 6 for smoothed posterior densities using the R "density()" function (R Core Development Team (2004)) from these draws. Note that Figure 6 show how informative the data are because the prior densities (not shown) are essentially flat or uniform in the regions displayed.

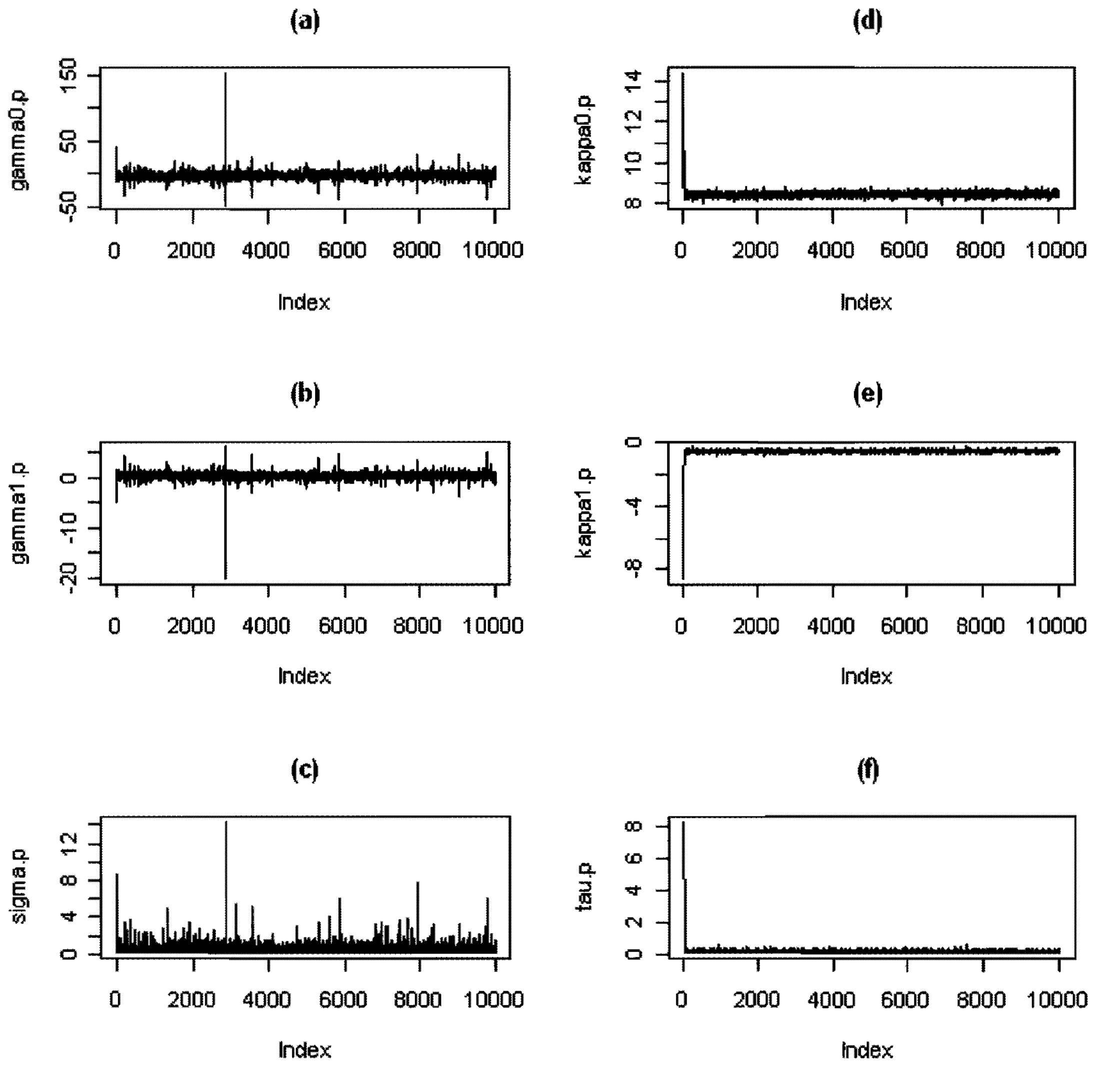

Figure 5: Traces of posterior draws for (a) γ_0, (b) γ_1, (c) σ, (d) κ_0, (e) κ_1, and (f) τ.

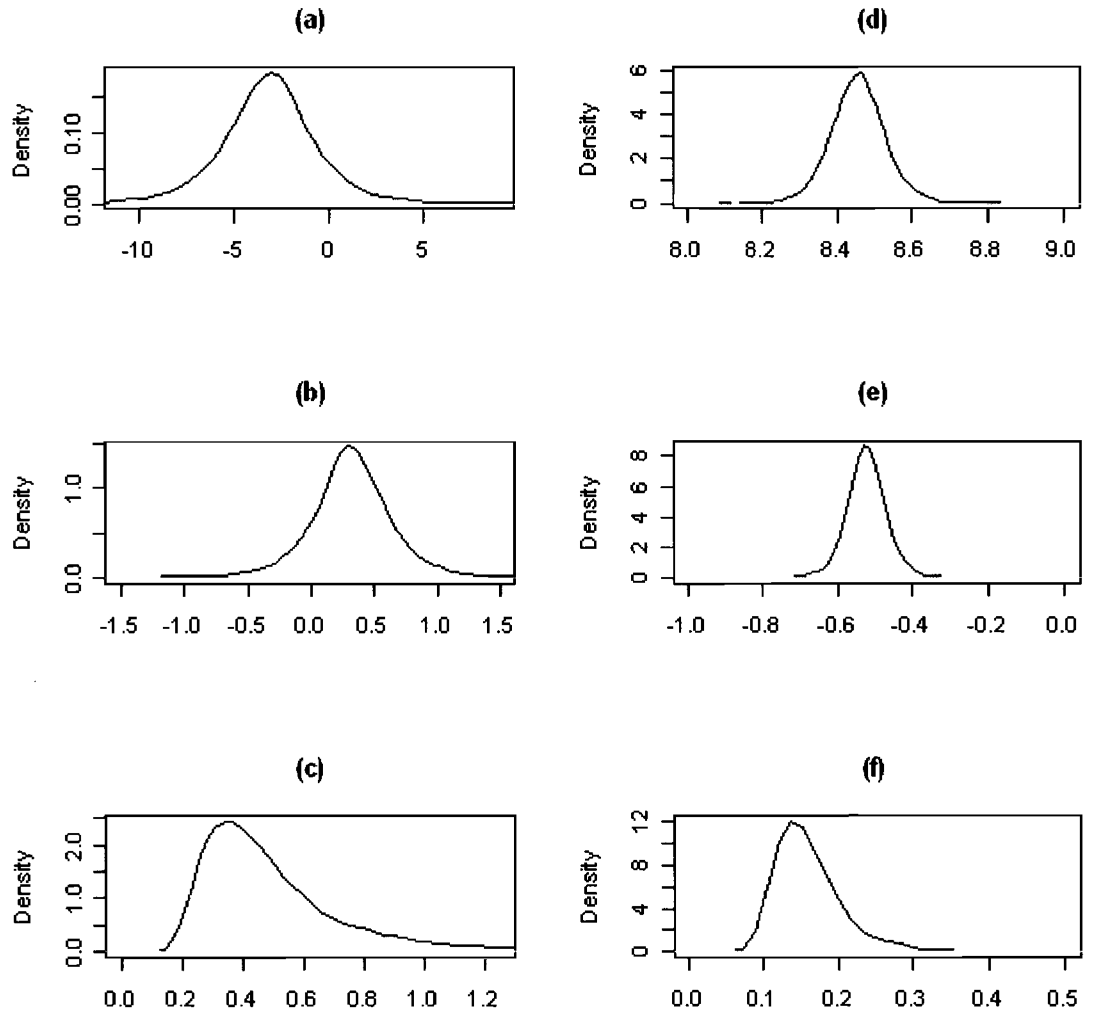

Figure 6: Smoothed densities of posterior draws for (a) γ_0, (b) γ_1, (c) σ, (d) κ_0, (e) κ_1, and (f) τ.

Prediction

We next consider several kinds of prediction. We begin with the prediction of performance y at a given CPP value x. From Equation 4, we have the computer code biased corrected performance model at a given CPP value x when we know $(\gamma_0, \gamma_1, \sigma^2)$ exactly. In this case, we only know $(\gamma_0, \gamma_1, \sigma^2)$ with uncertainty as described by the joint posterior distribution as characterized by the draws generated by WinBUGS described in the previous section. It turns out that the draws are actually easier to work with in calculating predictions. For a given x, repeatedly take a $(\gamma_0, \gamma_1, \sigma^2)$ draw, calculate $6.0 + 0.35x + \gamma_0 + \gamma_1 x$ and then draw a $Normal(6.0 + 0.35x + \gamma_0 + \gamma_1 x, \sigma^2)$ to obtain a draw from the y predictive distribution. Consequently, using the 10,000 draws of $(\gamma_0, \gamma_1, \sigma^2)$ generated by WinBUGS, we obtain 10,000 draws from the predictive distribution of performance y. See Figure 7 for the performance predictive distribution at $x = 7.5$ and $x = 8.5$ appearing as smoothed densities plotted as dashed lines. The true performance y densities using the exact values of $(\gamma_0, \gamma_1, \sigma^2)$ are plotted as solid lines. Note there is little difference between the predictive and true densities for these values of x where the performance data were taken.

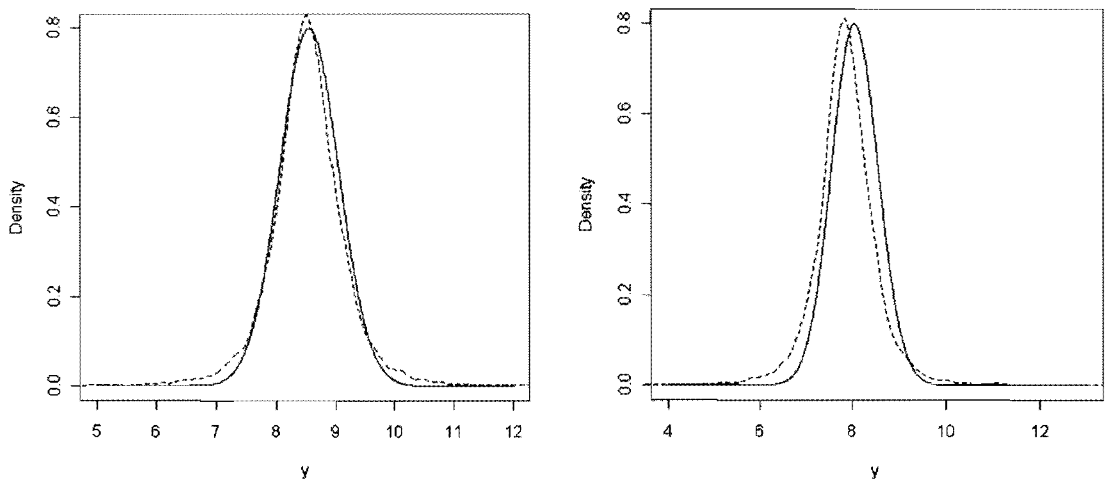

Figure 7: Performance y predictive distribution (dashed line) at x=8.5 on the left and x=7.5 on right with true performance distribution as a solid line.

In contrast, see Figure 8, which displays the performance predictive distribution at $x = 3$ appearing as the smoothed density plotted as a dashed line that is much more disperse and shifted slightly to the right of the true performance density plotted as a solid line. The shift results from the bias correction of the computer code not being perfect. The increased spread of the predictive distribution results in part from the uncertainty of the parameters $(\gamma_0, \gamma_1, \sigma^2)$, but mostly from extrapolating at $x = 3$, which is far from where the performance data were taken (7 to 9 as listed in Table 1).

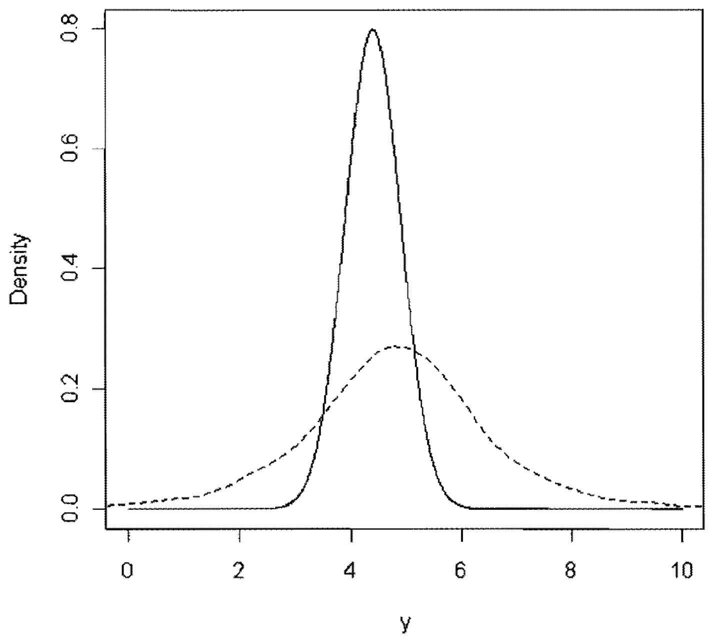

Figure 8: Performance y predictive distribution (dashed line) at x=3 with true performance distribution as a solid line.

Over the CPP x range from 3 to 8.5, we see in Figure 9 a plot of the 0.025, 0.5 (median), and 0.975 quantiles of the true performance distribution as solid lines. That is, between the upper and lower solid lines contain 95% of the true performance distribution. Also, a plot of the 0.025, 0.5 (median), and 0.975 quantiles of the predictive performance distribution are displayed as dashed lines. Note how the predictive distribution becomes more dispersed in regions away from where the performance data were collected and the bias (comparing the middle dashed and solid lines) resulting from the imperfect bias correction of the computer code. Also note that the computer code appears as the dotted line.

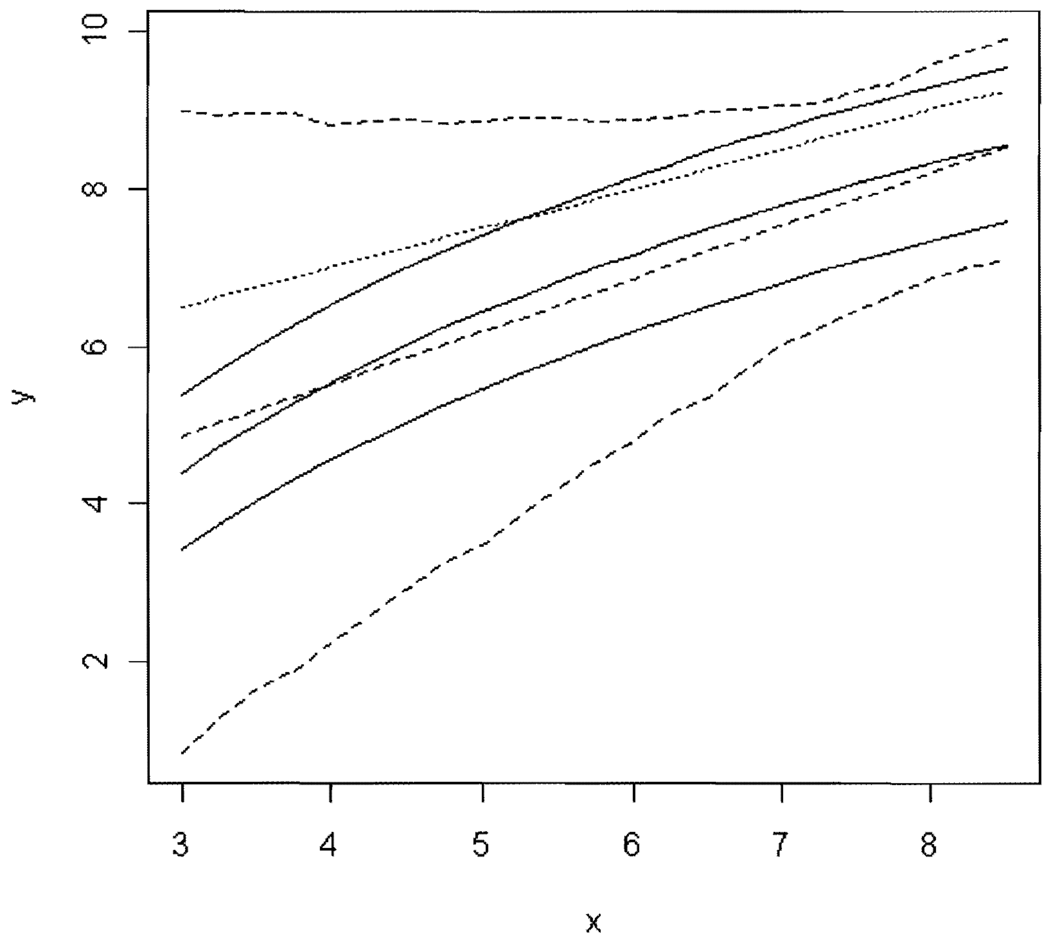

Figure 9: 0.025, 0.5, and 0.975 quantiles of the true performance distribution as solid lines, those of the predictive performance distribution as dashed lines and the computer code as a dotted line.

Similarly, for the predictive distribution of the CPP x population distribution at a given t, based on Equation 5, repeatedly take a $(\kappa_0, \kappa_1, \tau^2)$ draw, calculate $\kappa_0 + \kappa_1 t$ and then draw a $Normal(\kappa_0 + \kappa_1 x, \tau^2)$ to obtain a draw from the x predictive distribution. Consequently, using the 10,000 draws of $(\kappa_0, \kappa_1, \tau^2)$ generated by WinBUGS, we obtain 10,000 draws from the predictive distribution of CPP x. See Figure 10 for the performance predictive distribution at t = 0 and t = 2 appearing as smoothed densities plotted as dashed lines. The true CPP x densities using the exact values of $(\kappa_0, \kappa_1, \tau^2)$ are plotted as solid lines. Note there is little difference between the predictive and true densities for these values of t where the CPP data were taken. Contrast this with Figure 11, which displays the CPP predictive distribution at t = 11 appearing as the smoothed density plotted as a dashed line that is much more disperse and shifted slightly to the left of the true CPP density plotted as a solid line. The increased spread of the predictive distribution results in part from the uncertainty of the parameters $(\kappa_0, \kappa_1, \tau^2)$, but mostly from extrapolating at t = 11, which is far from where the CPP data were taken (0 and 2 as listed in Table 2).

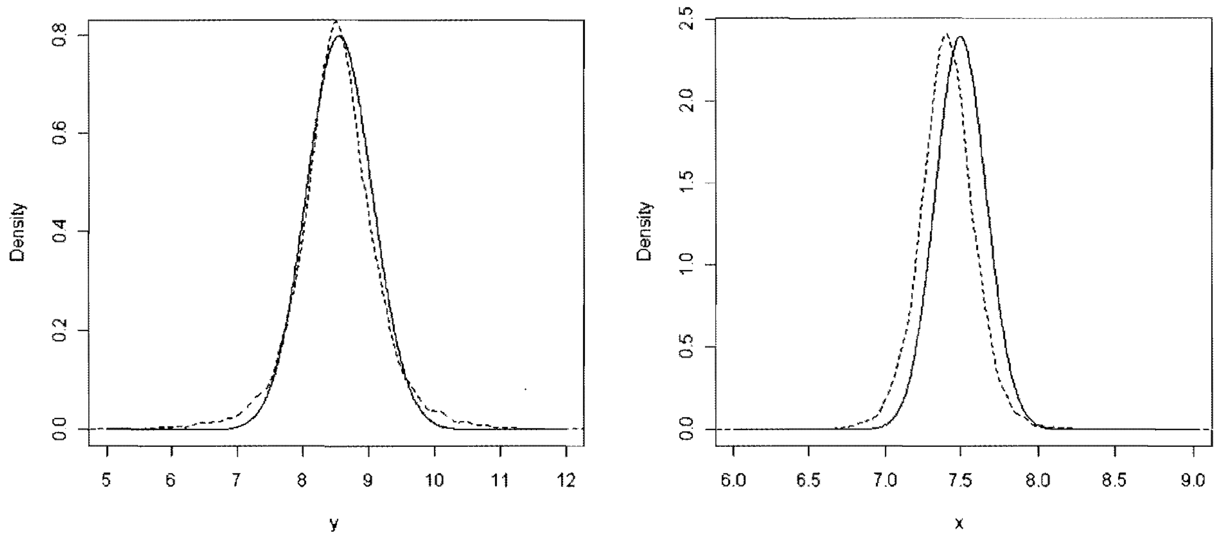

Figure 10: CPP x predictive distribution at t=0 on the left and t=2 on the right as dashed lines with true distribution as solid lines.

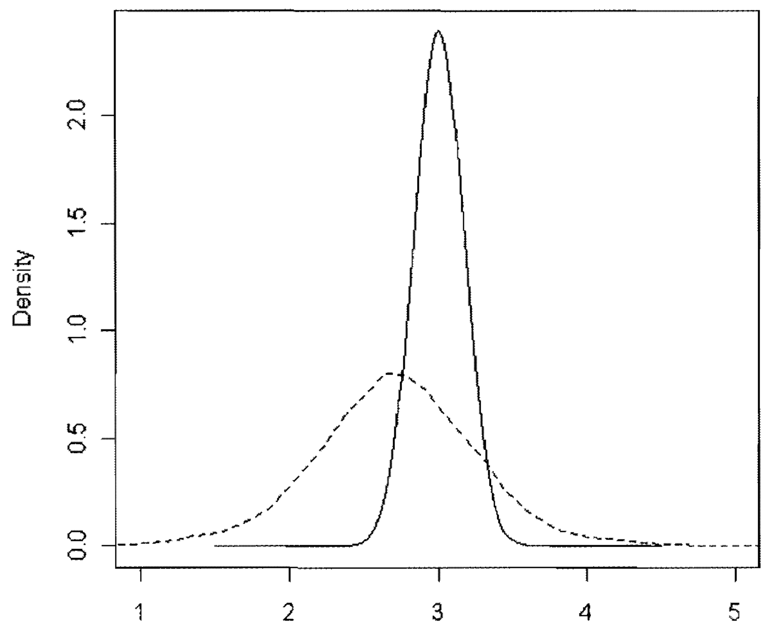

Figure 11: CPP x predictive distribution at t=11 as a dashed line with true distribution as a solid line.

Now, we can put this all together by generating the predictive distribution of the population performance y at various times t. By the population performance y distribution, we mean the CPP x population distribution propagated through the performance y distribution. That is, we obtain a sample by first drawing an x according to Equation 4 and then drawing a y according to Equation 5. These true performance y population distributions at times t = 0, 2, and 11 appear in Figures 12 and 13 as solid lines. We obtain various versions of predictive distributions by using the predictive distribution of the CPP x population and/or the predictive distribution of the performance y. The dashed line is the predictive distribution obtained from both the predictive performance and CPP population distributions. That is, we take draws from the joint posterior distributions of $(\kappa_0, \kappa_1, \tau^2)$ and $(\gamma_0, \gamma_1, \sigma^2)$. Then, we use these posterior values in Equations 4 and 5 to obtain a draw from the predictive distribution. The dotted line is the predictive distribution from the predictive performance distribution and true CPP population distribution. Finally, the dotted-dashed line is the predictive distribution from true performance distribution and the predictive CPP population distribution.

In Figure 12, we see little difference between the true population performance distribution and the various predictive distributions at times $t = 0$ and $t = 2$. The predictive distributions, which account for different scenarios of uncertainty, are necessarily more dispersed, i.e., more uncertain. Recall that the predictions are made where the performance and CPP data were taken. In Figure 13, there is a more striking difference between the true population performance distribution and the various predictive distributions at times $t = 11$, which is far from where the performance and CPP data were taken. Note the dramatic increase in uncertainty by using the predictive performance distribution as seen by the dashed and dotted lines. Also notice that the predictive CPP population distribution does not add much to the overall uncertainty (dashed line versus the dotted line) when the predictive performance distribution is used.

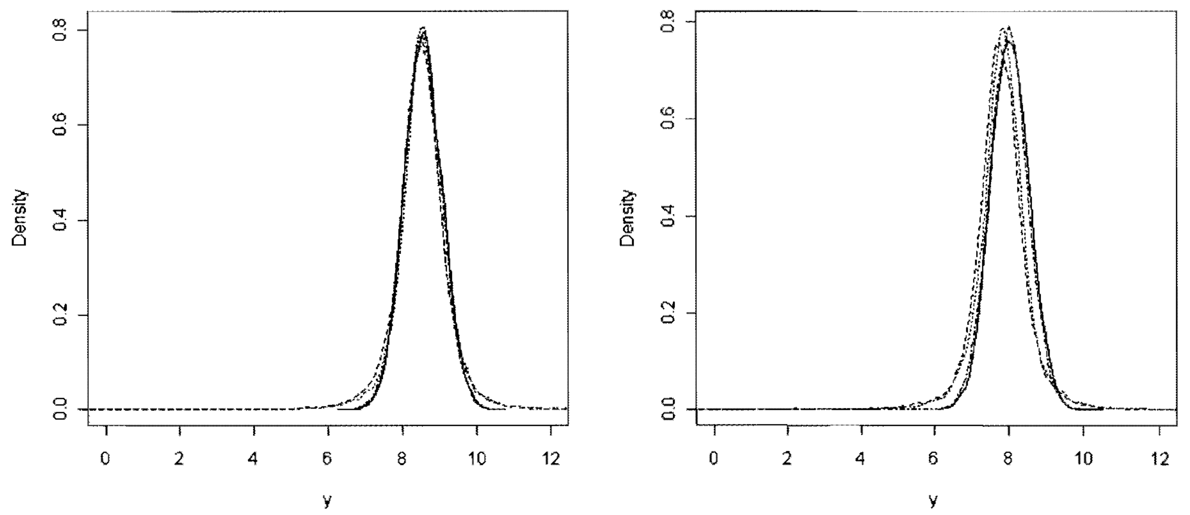

Figure 12: Predictive distributions of performance y distribution of population at t=0 on the left and t=2 on the right: solid line is true performance distribution, dashed line is predictive distribution from both estimated performance and CPP population distributions, dotted line is predictive distribution from estimated performance distribution and true CPP population

distribution, dotted-dashed line is predictive distribution from true performance distribution and estimated CPP population distribution.

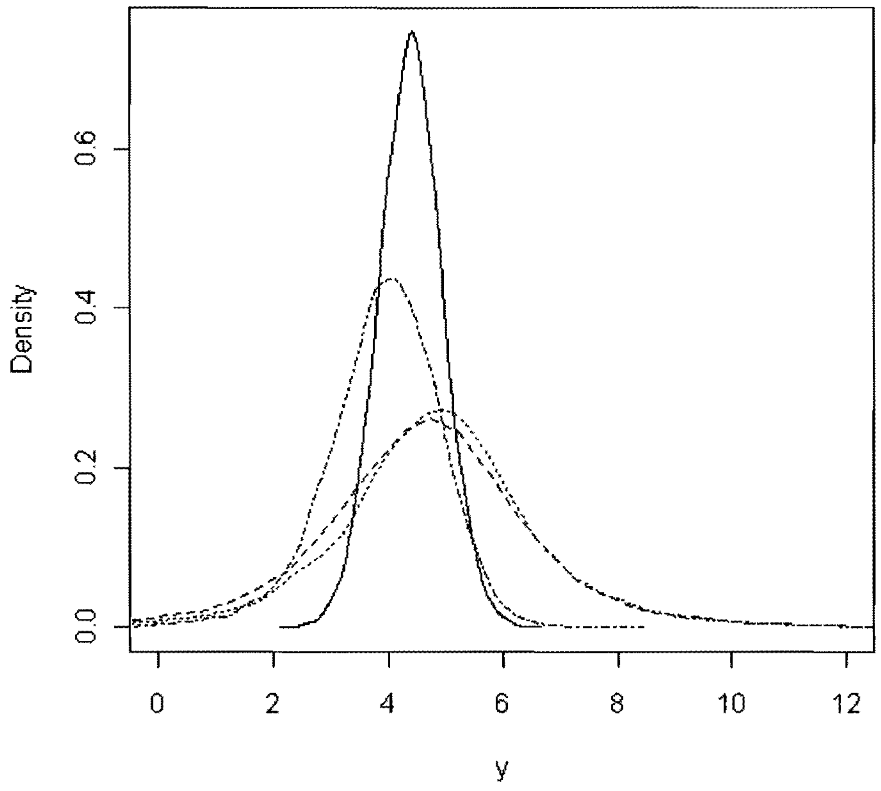

Figure 13: Predictive distributions of performance y distribution of population at t=11: solid line is true performance distribution, dashed line is predictive distribution from both estimated performance and CPP population distributions, dotted line is predictive distribution from estimated performance distribution and true CPP population distribution, dotted-dashed line is predictive distribution from true performance distribution and estimated CPP population distribution.

Over time t from 0 to 11 years, we see in Figure 14 a plot of the 0.025, 0.5 (median), and 0.975 quantiles of the true performance distribution as solid lines. That is, between the upper and lower solid lines contain 95% of the true performance distribution. Also, a plot of the 0.025, 0.5 (median), and 0.975 quantiles of the predictive performance distribution obtained by propagating the predictive CPP distribution are displayed as dashed lines. Note how the predictive distribution becomes more dispersed in regions away from where the CPP and performance data were collected and the bias (comparing the middle dashed and solid lines) resulting from the imperfect bias correction of the computer code.

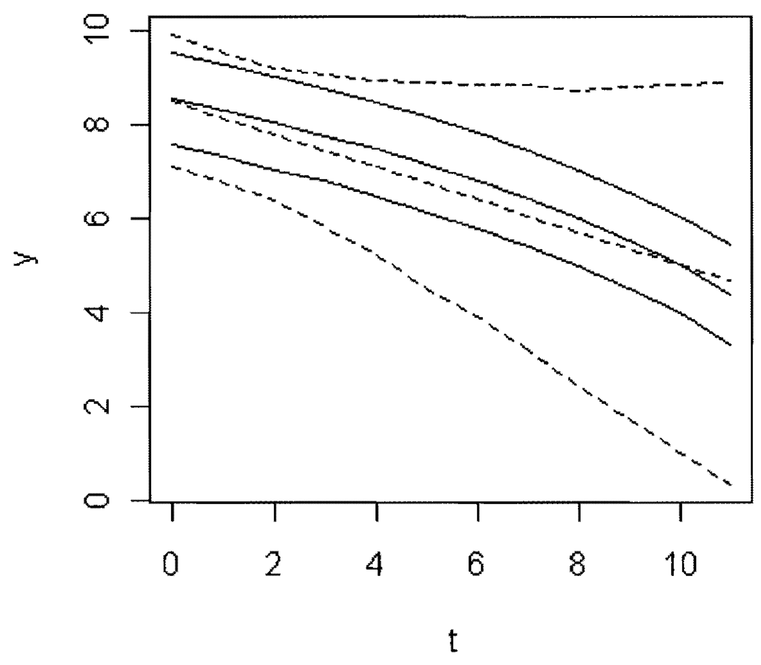

Figure 14: 0.025, 0.5, and 0.975 quantiles of the true performance distribution y as solid lines and those of the predictive performance distribution as dashed lines over time t in years.

Discussion

In this paper, we have provided a generic example to illustrate various points about making future predictions of population performance when there are a biased performance computer code, actual performance data, and critical performance parameter data sampled from the population at various times. Some of the key aspects of the illustration is how the actual performance data help to correct the biased computer code and the impact of uncertainty especially when the prediction is made far from where the available data were taken. Finally, we demonstrated how a Bayesian approach allows both inferences about the unknown parameters and predictions to be made in a consistent framework, i.e., probability distributions are propagated through subsequent probability distributions.

We believe that the generic example exhibits the features found in real applications. In real applications, there are likely multiple performance parameters, whose population joint distribution can be characterized by a multivariate probability distribution. The computer code itself may need to be modeled nonparametrically when running the computer code is expensive. Nevertheless, both the computer code and discrepancy can be modeled using Gaussian stochastic processes (Kennedy and O'Hagan (2001)).

Finally, the generic example can also be carried further to assess the impact of the additional information. For example, if more data can be taken, where should they be taken and what is their impact on reducing uncertainty.

Acknowledgement

We thank C.C. Essex for her support and encouragement of this work.

References

Casella, G., George, E. (1992). Explaining the Gibbs Sampler. *The American Statistician*, 46: 167-174.

Chib, S., Greenberg, E. (1995). Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49: 327-335.

DeGroot, M.H. (1970). *Optimal Statistical Decisions*. New York, NY: McGraw-Hill.

Gelfand, A.E., Smith, A.F.M. (1990). Sampling-Based Approaches to Calculating Marginal Densities. *Journal of the American Statistical Association*, 85: 398-409.

Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B. (2003). *Bayesian Data Analysis, Second Edition*. Boca Raton, FL: Chapman & Hall/CRC.

Kennedy, M., O'Hagan, A. (2001). Bayesian Calibration of Computer Models (with discussion). *Journal of the Royal Statistical Society, Series B*, 63: 425-464.

R Development Core Team (2004). R: A Language and Environment for Statistical Computing. Vienna: R Foundation for Statistical Computing. (http://www.R-project.org).

Spiegelhalter, D., Thomas, A., Best, N., Lunn, D. (2004). *WinBUGS Version 1.4 User Manual.*

Appendix WinBUGS Code

WinBUGS code for performance data

```
model
{
        for( i in 1 : N ) {
        junk[i]<-ind[i]
                y.p[i] ~ dnorm(mu[i],inversesigma2)
                mu[i] <- 6+0.35*x.p[i]+gamma0 + gamma1*x.p[i]
        }
        inversesigma2 ~ dgamma(0.001,0.001)
        sigma <- 1 / sqrt(inversesigma2)
        gamma0 ~ dnorm(0.0,1.0E-6)
        gamma1 ~ dnorm(0.0,1.0E-6)
}
```

Data

```
list(N = 5)
```

Inits

```
list(gamma0=0,gamma1=0, inversesigma2 = 1)
```

WinBUGS performance data input file

```
ind[]  x.p[]   y.p[]
1 7.0 7.428685
2 7.5 8.314868
3 8.0 7.865882
4 8.5 8.283223
5 9.0 9.1252
END
```

WinBUGS code for performance data

```
model
{
        for( i in 1 : N ) {
        junk[i]<-ind[i]
                x.0[i] ~ dnorm(lambda.0[i],inversetau2)
                lambda.0[i] <- kappa0+kappa1*t.0[i]
        }
        for( i in 1 : N ) {
                x.2[i] ~ dnorm(lambda.2[i],inversetau2)
                lambda.2[i] <- kappa0+kappa1*t.2[i]
        }
```

```
                inversetau2 ~ dgamma(0.001,0.001)
                tau <- 1 / sqrt(inversetau2)
                kappa0 ~ dnorm(0.0,1.0E-6)
                kappa1 ~ dnorm(0.0,1.0E-6)
        }
Data
        list(N = 5)
Inits
        list(kappa0=0,kappa1=0, inversetau2 = 1)
```

WinBUGS CPP data input file

```
ind[]    x.0[]    x.2[] t.0[] t.2[]
1 8.418151 7.195434 0 2
2 8.445346 7.506274 0 2
3 8.640867 7.639724 0 2
4 8.453777 7.375526 0 2
5 8.323710 7.346231 0 2
END
```