



All Theses and Dissertations

2011-06-01

A Bayesian Approach to Missile Reliability

Taylor Hardison Redd
Brigham Young University - Provo

Follow this and additional works at: <https://scholarsarchive.byu.edu/etd>

 Part of the [Statistics and Probability Commons](#)

BYU ScholarsArchive Citation

Redd, Taylor Hardison, "A Bayesian Approach to Missile Reliability" (2011). *All Theses and Dissertations*. 2733.
<https://scholarsarchive.byu.edu/etd/2733>

This Selected Project is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in All Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.

A Bayesian Approach to Missile Reliability

Taylor Hardison Redd

A Project submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of
Master of Science

Dr. C. Shane Reese, Chair
Dr. Gilbert W. Fellingham
Dr. William F. Christensen

Department of Statistics
Brigham Young University

August 2011

Copyright © 2011 Taylor Hardison Redd

All Rights Reserved

ABSTRACT

A Bayesian Approach to Missile Reliability

Taylor Hardison Redd
Department of Statistics, BYU
Master of Science

Each year, billions of dollars are spent on missiles and munitions by the United States government. It is therefore vital to have a dependable method to estimate the reliability of these missiles. It is important to take into account the age of the missile, the reliability of different components of the missile, and the impact of different launch phases on missile reliability. Additionally, it is of importance to estimate the missile performance under a variety of test conditions, or modalities. Bayesian logistic regression is utilized to accurately make these estimates. This project presents both previously proposed methods and ways to combine these methods to accurately estimate the reliability of the Cruise Missile.

Keywords: Bayesian reliability, Bayesian logistic regression, Metropolis-Hastings, Gibbs sampling

ACKNOWLEDGMENTS

This project would not be possible without the help, encouragement, and support from numerous friends, family members, and advisors. To all of you, thank you. Specifically, I would like to thank my parents: my father, Hugh, for being such an example and source of advice, and my mother, Cindy, for her love, support, and encouragement. My education would not have been possible without you. I would also like to thank my siblings, Jessica, Trevor, Preston, Dillon, and Rebecca, for all their support throughout my life. Thank you, Trevor, for editing this work and making more corrections than one can count and Preston for all his help throughout school. Finally, I would like to thank two very special advisors—Dr. Jeffrey Humpherys for introducing me to research and the opportunities that mathematics provides and Dr. C.Shane Reese for taking a chance on me. Thank you for allowing me to work with you and learn so much. Once again, thank you all for your support.

CONTENTS

Contents	vii
1 Missile Reliability	1
1.1 Introduction	1
1.2 Missile Specifics	1
2 Literature Review	3
2.1 Introduction	3
2.2 Bayesian Reliability	3
2.3 Multiple Components	4
2.4 Different Modality in Reliability	5
2.5 Aging	7
2.6 Methodology for Generating Posteriors	8
2.7 Model Fit	9
2.8 Model Selection	11
3 Methods and Proposal	17
3.1 Model 1	17
3.2 Model 2	20
3.3 Model 3	20
3.4 Test Cases	21
3.5 Model Selection	26
4 Evaluation of the Models	29

4.1	Introduction	29
4.2	MCMC Algorithm	29
4.3	Success Criteria	30
4.4	Model Comparison and Discussion	43
5	Conclusions	47
5.1	Model Selection	47
5.2	Further Work	47
	Bibliography	51
	Appendices	55
	Appendix A: MATLAB CODE	57
	A.1 Main Program	57
	A.2 Likelihood function	75

MISSILE RELIABILITY

1.1 INTRODUCTION

Each year the United States Government spends billions of dollars on military expenses. A large portion of that spending accounts for missiles and munitions. It is estimated that in 2011 approximately \$12.9 billion will be spent in this sector alone. With so much money spent in this area, it is important to know exactly which munitions are available and determine with what degree of certainty these munitions will work. For this reason reliability of these missiles and munitions is so important. The goal of this project is to develop a model for and methods of determining the reliability of the Cruise Missile.

1.2 MISSILE SPECIFICS

The missile being modeled is consists of 41 components, working in series, that go through four different phases. The goal is to estimate the reliability of the missile—possibly by calculating the reliability of each individual component in each phase. Knowing the reliability of each component, in turn, allows us to compute our estimated total reliability.

The main goal is to calculate the underlying reliability of each missile; however, there are certain effects that we would like to take into account. First, we would like to account for an aging effect. We would like to be able to allow the reliability to change depending on the age of the missile. One way to approach the effects of aging is to have a single effect of age on the system. If, however, we would like to allow for greater flexibility, we could allow each component to have its own aging effect.

The other effect we would like to model is the test conditions. The missile is tested in three different test conditions. While only one test condition is similar to the conditions the missile will actually be fired in, the other two conditions are still useful in estimating reliability. We would like to find a model that can take into account these different test conditions, or modalities, when estimating the reliability of the missile.

LITERATURE REVIEW

2.1 INTRODUCTION

Loosely stated, reliability is the probability of a component to successfully perform its intended function under certain circumstances. This, however, is a vague description that neglects questions that need to be considered: at what standard does it work, under what conditions will it work, and how long will it work at that standard and under those conditions.

For example, it may be reasonable to define reliability as the probability of a device functioning properly. Alternatively, one might be interested in failure time, defined as the time an object, such as a light bulb, will work (Hamada, Wilson, Reese, and Martz 2008). While both these definitions of analysis are important for different types of problems, we will concentrate our focus on the first example. Specifically, we are only interested in the probability of an item (the missile or component) accomplishing a task at a given time. No attention will be paid to lifetime or failure time analysis or to the performance of the missile. We will only look at the outcome as a success or failure. Therefore, reliability is defined here as the probability of a success.

2.2 BAYESIAN RELIABILITY

In this analysis, we use Bayesian methods to estimate reliability. Bayesian analysis consists of two parts: the likelihood function and the prior distributions. The likelihood function is the sampling distribution of the data. Contained in the likelihood function are parameters that are treated as unknown quantities. Probability densities are assigned to each parameter to describe its uncertainty. These densities are

referred to as prior distributions. It is here that expert opinion can be incorporated to help define the priors.

Bayesian methods are preferred over classical maximum likelihood estimates for numerous reasons. First, Bayesian methods allow us a formal way to incorporate expert knowledge when estimating reliability. This will be important as we have little knowledge of missiles or missile reliability. Second, in the presence of scarce data, Bayesian methods allow for much narrower interval estimates than classical inference. This is due to the information obtained by the prior distribution. Finally, Bayesian methods allow for increased flexibility when modeling reliability. As a brief example, consider the case where there are 10 missile tests and all tests succeed. The MLE, maximum likelihood estimate, of the missile reliability would be an unrealistic value of 1. Using Bayesian methods, our posterior distribution for the reliability would not be 1. In fact, we would not have a point estimate for the reliability. Instead, we would have a distribution that would not have all probability mass at 1. This allows a much more realistic estimate.

2.3 MULTIPLE COMPONENTS

In the most basic case, we can model system reliability as

$$X_S \sim \text{Binom}(n, R_S),$$

$$R_S \sim \text{Beta}(\alpha, \beta),$$

where X_S is the number of successful tests of the system, n is the number of tests, and R_S is the probability of a successful test, or system reliability. This specific model eliminates the problem of occasionally getting unrealistic estimates that are generated from classical reliability methods while also allowing us to incorporate expert opinion in the prior. However, it does a poor job in allowing flexibility to model things such as multiple components, different modality, and aging effects.

In response to having multiple components and subsystems, Martz, Waller, and Fickas (1988) and Martz and Waller (1990) also proposed using binomial likelihoods and beta priors. However, instead of having a likelihood on the system outcome and a prior on the system reliability, R_S , each component and subsystem outcome (for example, component i) would have its own likelihood and prior for R_i , where R_i indicates the reliability of the i^{th} component. Since the system relies on these components and subsystems, these likelihoods and priors would then induce a posterior reliability on the system. For example, assume we have a system with three components in series. Each of these three components would be given their own likelihood and prior. This would lead to the following model:

$$X_1 \sim Binom(n, R_1),$$

$$X_2 \sim Binom(n, R_2),$$

$$X_3 \sim Binom(n, R_3),$$

$$R_1 \sim Beta(a_1, b_1),$$

$$R_2 \sim Beta(a_2, b_2),$$

$$R_3 \sim Beta(a_3, b_3),$$

where n is the number of tests, X_i is the number of successes on the i^{th} component, and R_i is the reliability of the i^{th} component. Assuming three components independently in series gives rise to an induced system reliability, $R_S = R_1 * R_2 * R_3$.

2.4 DIFFERENT MODALITY IN RELIABILITY

Missiles are often tested using different procedures. For example, some tests are nondestructive, others are destructive. These different types of tests, or modalities, lead to the missile going through more severe conditions. To take these different

conditions into account, Robinson and Chang (2003) allowed each different mode to have its own reliability. They proposed the model,

$$X_k \sim \text{Binomial}(n_k, R_k),$$

$$R_k \sim \text{Beta}(\alpha, \beta),$$

$$\alpha \sim \text{Weibull}(\nu, \lambda),$$

$$\beta \sim \text{Uniform}(0, 10),$$

where k refers to the different test modality, and X_k is the number of successes in the n_k trials for the k^{th} modality with reliability R_k . This allows the reliability to change as a function of the modality. While this is often a good model to use, there are situations when it creates difficulties.

Reese, Hamada, and Robinson (2005) noted that the model produced by Robinson and Chang assumed the reliabilities, R_k , are conditionally independent. Essentially, each modal reliability comes from a ‘population’ of reliabilities for all test modalities. This assumption makes each reliability related; however, it also introduces more variation in the results than assuming a single reliability across modalities. This increase in variation is especially evident in modalities with few test cases. Without a large number of test cases, it is reasonable to have more variation in the posterior reliability. However, this increased variation in the posterior of the reliability for modalities with few test cases is then passed across or propagated to the other modalities. This occurs because they are assumed to come from the same population of reliabilities.

To combat these difficulties, Reese et al. (2005) proposed using an extension of logistic regression in order to take into account these different modalities and conditions. Their motivation was that “all test modalities are attempting to estimate the same quantity, only that some may have less fidelity than others. More precisely, tests with more fidelity are ones which agree more closely with actual test conditions.”

Let R be the reliability of the missile in the actual test conditions. Conditions that are harsher than actual test conditions will decrease this underlying reliability, R ; while softer conditions will increase it. Therefore, they suggest estimating R while simultaneously estimating the effect of the different conditions.

To do this, they propose taking the logit transform of the ‘base’ reliability, R . This reliability will serve as a point of reference for the other modalities. Expressing R on the logit scale, we have

$$\text{logit}(R) = \log\left(\frac{R}{1-R}\right) = \mu. \quad (2.1)$$

Now, instead of estimating R , we will estimate the quantity of interest on the logit scale, μ . With this point of reference, we can now express R_k , where k indicates the modality, on the logit scale as

$$\text{logit}(R_k) = \log\left(\frac{R_k}{1-R_k}\right) = \mu + \lambda_k. \quad (2.2)$$

This allows all tests, regardless of modality, to influence estimates of our base reliability, R . Once these estimates are obtained, we can then convert them back to the reliability using the inverse logit relationship,

$$R = \frac{\exp(\mu)}{1 + \exp(\mu)}, \quad (2.3)$$

to calculate R . Similarly, to calculate R_k we use the inverse logit relationship

$$R_k = \frac{\exp(\mu + \lambda_k)}{1 + \exp(\mu + \lambda_k)}. \quad (2.4)$$

2.5 AGING

Because there is often a degradation of materials with time, we would like to be able to find the effect of aging on the reliability of the missile. As stated before, (2.2) leads us to logistic regression on the reliability. The terms μ and λ_k are explanatory variables for the reliability. While the above equation only takes into account the

modality of the test, we can expand it to take into account the age of the missile. This extension leads to

$$\log\left(\frac{R_i}{1 - R_i}\right) = \mu + \lambda_i + \alpha * age, \quad (2.5)$$

where α is the effect of age on the reliability of the missile. This allows for a very flexible model that can easily incorporate the effect time has on the missile.

2.6 METHODOLOGY FOR GENERATING POSTERIORES

Once we have specified a model, we turn to generating posterior distributions for the unknown parameters and hyperparameters. Generating draws from the joint posterior can often be difficult as the dimension of the parameter space becomes large. Therefore, it is usually desired to use the marginal distributions to generate random variables. In practice, however, obtaining these distributions is extremely difficult. One method to get around actually calculating the marginal distributions is Gibbs Sampling. While the method was mostly developed by Metropolis, Rosenbluth, Rosenbluth, Teller, and Teller (1953) and further developed by Hastings (1970), it has recently received interest due to Gelfand and Smith (1990).

Essentially, we first find the distribution of a given parameter or hyperparameter, conditional on the rest of the parameters and hyperparameters. We then take draws from these conditional distributions, or complete conditionals. These draws are then used as values in the other complete conditionals. Iterating through each parameter/hyperparameter for each draw and taking a sufficiently large number of draws will result in characteristics of the marginal densities of each parameter. Casella and George (1992) give a more detailed explanation of Gibbs sampling.

If, when calculating the complete conditional, the form of a known distribution is found, draws can be taken from a random number generator for that distribution. On the other hand, if the complete conditional, denoted $g(\theta)$, does not take the form

of a known distribution, the Metropolis-Hastings algorithm can be implemented. This algorithm was also developed in the papers by Metropolis et al. (1953) and Hastings (1970). In the Metropolis-Hastings algorithm, we start with an initial value, θ_0 , of a parameter. We then generate a new ‘candidate’ value, θ_N , for a draw from the posterior. This candidate value is then accepted as a draw from the posterior with probability $\frac{g(\theta_0)}{g(\theta_N)}$. If this value is not accepted, the old value is taken as the new draw. For a more detailed description of the Metropolis-Hastings algorithm, see Chib and Greenberg (1995).

2.7 MODEL FIT

As we continue to adjust and to try to improve the model by correctly accounting for different influencing factors, it is important to make sure we are accurately modeling the reliability of the missile. It is possible that as we create our model we do not do an adequate job of modeling the reliability. In order to test how well we do in modeling the reliability we turn to the Bayesian χ^2 Test. This test allows us to test model fit.

The Bayesian χ^2 test is motivated by the χ^2 goodness-of-fit diagnostic that was first proposed by Pearson (1900). As this is a common test we use the definition given by Hamada, Wilson, Reese, and Martz (2008). “Let y_1, \dots, y_n denote an independent and identically distributed sample drawn from distribution $f(y|\boldsymbol{\theta})$ ” where $\boldsymbol{\theta}$ is known. “Let $0 = a_0 < a_1 < \dots < a_{K-1} < a_K = 1$ denote predetermined quantiles from a uniform distribution, and define $p_j = a_j - a_{j-1}$. Finally, let m_j denote the number of observations y_i for which $a_{j-1} < F(y_i|\boldsymbol{\theta}) \leq a_j$. Then Pearson’s chi-squared test statistic, R^0 , may be defined as

$$R^0 = \sum_{k=1}^K \frac{(m_k - np_k)^2}{np_k}.” \tag{2.6}$$

Pearson (1900) showed that R^0 follows a chi-squared distribution with $K - 1$ degrees of freedom when the model is true. Note, this is true when $\boldsymbol{\theta}$ is known. Cramér

(1946) and Chernoff and Lehmann (1954) address when $\boldsymbol{\theta}$ is estimated using different methods.

Johnson (2004) loosened the standard assumptions in two ways to define a Bayesian version of the χ^2 goodness-of-fit statistic. First, instead of using the ‘true’ or optimal value of $\boldsymbol{\theta}$, as in (2.6), we will use a randomly sampled draw of $\boldsymbol{\theta}$ from the posterior. This randomly sampled draw, $\tilde{\boldsymbol{\theta}}$, is then used to estimate p_k . The assumption of identically distributed data is relaxed by redefining the bin counts m_j according to the number of observations y_i for which $a_{j-1} < F_i(y_i|\tilde{\boldsymbol{\theta}}) \leq a_j$, where $F_i(y_i|\tilde{\boldsymbol{\theta}})$ is the conditional distribution function for the sampled value of $\tilde{\boldsymbol{\theta}}$. Therefore, a Bayesian version of (2.6) can be defined as

$$R^B(\tilde{\boldsymbol{\theta}}) = \sum_{k=1}^K \frac{[m_k(\tilde{\boldsymbol{\theta}}) - np_k]^2}{np_k}. \quad (2.7)$$

For large n , the sample size, the distribution of R^B is χ_{K-1}^2 , independently of the dimension of $\boldsymbol{\theta}$ (Johnson 2004). This allows for two important features of the statistic. First, there does not need to be any adjustment for the dimensionality of $\boldsymbol{\theta}$. Second, the statistic allows for nonidentically distributed data. These features allow the statistic to be used in a much broader range of models than the χ^2 statistics proposed by Pearson (1900), Cramér (1946), and Chernoff and Lehmann (1954).

It is important to note, R^B is based off of one randomly selected draw from the posterior, $\boldsymbol{\theta}$. In response to this, the average value of R^B over many draws from the posterior, can be found; however, this averaged value has no known reference distribution (like χ_{K-1}^2 in the case of a single value). Alternatively, we could report the proportion of values that exceed a critical value of the distribution χ_{K-1}^2 (Hamada et al. 2008). Other methods for assessing significance for values of R^B are discussed by Johnson (2007).

2.8 MODEL SELECTION

Once we find multiple models to use, it is also important to have a formal way to compare these models. This will allow decisions to be made on which model and set of alterations should be used as we move forward. While there are many different approaches to model selection, we will focus on four: Bayes Factor, Bayesian Information Criterion (BIC), Akaike Information Criterion (AIC), and Deviance Information Criterion (DIC). These approaches are discussed below.

Bayes Factor

One of the first and most basic, conceptually, of model selection tools is the use of a Bayes Factor. Bayes Factor's allow for comparison by essentially finding the odds that a given model is the correct model. For a given model, M_1 , we can find the odds with

$$\frac{Pr(M_1|D)}{1 - Pr(M_1|D)},$$

where D indicates the data. Given two different models to compare, we can simplify this, using Bayes' Theorem, to

$$B_{12} = \frac{Pr(D|M_1)Pr(M_1)}{Pr(D|M_2)Pr(M_2)}, \quad (2.8)$$

where B_{12} is the Bayes Factor between model 1 and model 2. Typically, the prior odds, $\frac{Pr(M_1)}{Pr(M_2)}$, are equal to 1. This then reduces to

$$\frac{Pr(D|M_1)}{Pr(D|M_2)} = \frac{\int Pr(D|\Theta_1, M_1)\pi(\Theta_1|M_1)d\Theta_1}{\int Pr(D|\Theta_2, M_2)\pi(\Theta_2|M_2)d\Theta_2}. \quad (2.9)$$

Essentially, the Bayes Factor is the likelihood ratio hypothesis test except for a few key differences:

- The specification of model 1 to model 2 does not matter.

- The likelihood is integrated over Θ , as opposed to maximized. This allows for comparison across the entire Θ space instead of just a point estimate.
- Model 1 and model 2 do not need to be nested models. They can differ with respect to likelihoods, priors, hyperpriors, etc.

The value $\log(B_{12})$ can be used to shed insight on which model to use. Table 2.1, shown below, summarizes the different values of $\log(B_{12})$ and its interpretation.

$\log(B_{12})$	Evidence in favor of M_1 over M_2
0 – 0.5	Not worth more than a bare mention
0.5 – 1.0	Substantial
1.0 – 2.0	Strong
> 2.0	Decisive

Table 2.1: Functional interpretation of natural logarithm of Bayes Factors for comparing two models, M_1 and M_2

Bayes' factors can often be difficult to calculate, especially as the model complexity increases. DiCiccio, Kass, Raftery, and Wasserman (1997) supply different methods and approximations for Bayes' factors. One approximation is given below in (2.11).

BIC

While the method of using Bayes Factors for model selection is very useful, one serious drawback is that it can often be difficult to calculate. Schwarz (1978) gives one possible solution to this difficulty with the Bayesian Information Criterion, or BIC, which is defined as

$$BIC = -2\log[f(y|\hat{\Theta})] + p\log n, \tag{2.10}$$

where p is the number of parameters of the model, n is the number of observations, and $f(y|\hat{\Theta})$ is the likelihood. As with most information criterion-based methods, models with a lower BIC than other models present evidence as the better model.

We can use the BIC value to find an approximation of the Bayes Factor (Carlin and Louis 2009). This approximation is given by

$$\widehat{B}_{12} = \exp \left\{ -\frac{1}{2}(BIC_1 - BIC_2) \right\}. \quad (2.11)$$

BIC is typically preferred over Bayes' factor because it is often used to approximate the Bayes' factor (Hamada et al. 2008).

While there are many ways to choose $\hat{\Theta}$ for $f(y|\hat{\Theta})$, often the posterior mean is used. If the posterior distribution is skewed, the posterior median can also be used. It is important to note that the posterior mean and median can depend greatly on the prior distributions of Θ for each model. A logical question is then, "How can one compare models without letting priors influence the decision?" Replacing $f(y|\hat{\Theta})$ with $\max_{M1} f(y|\Theta)$ takes the effect of priors out of the calculations. Essentially, maximizing the likelihood allows the different information criterion-based methods to be calculated with little to no influence from the priors.

AIC

Similar to BIC is the Akaike Information Criterion, or AIC. Akaike (1974) proposed the following information criterion-based method,

$$AIC = -2 \log [f(\mathbf{y}|\boldsymbol{\theta})] + 2p. \quad (2.12)$$

Like BIC, the first term in (2.12) is to account for model fit, while the second term is a penalty for the number of parameters included in the model. Similar to BIC, when using AIC, it is not the absolute size but the relative size—specifically, the difference between models—that is important (Burnham and Anderson 2002). While this is a

useful information criterion tool, many practitioners have noted that using AIC tends to keep too many terms in the model. This makes sense because the penalty for added parameters is relatively small when compared to $\log(n)$ (Carlin and Louis 2009).

DIC

Spiegelhalter, Best, Carlin, and van der Linde (2002) proposed another way of comparing models. This alternative is especially effective when comparing hierarchical models. While BIC penalizes for the number of parameters in a model, they note that “the inclusion of a prior distribution induces a dependence between parameters that is likely to reduce the effective dimensionality, although the degree of reduction may depend on the data that are available.” It is therefore important to be able to find an estimate for the effective number of parameters in a model. The effective number of parameters can be estimated by p_D , defined as

$$p_D = \overline{D(\boldsymbol{\theta})} - D(\bar{\boldsymbol{\theta}}), \quad (2.13)$$

where

$$D(\boldsymbol{\theta}) = -2 \log [f(\mathbf{y}|\boldsymbol{\theta})],$$

and $f(\mathbf{y}|\boldsymbol{\theta})$ is the likelihood function. $D(\boldsymbol{\theta})$ is considered the ‘Bayesian deviance,’ with $\overline{D(\boldsymbol{\theta})}$ being the mean of the deviances and $D(\bar{\boldsymbol{\theta}})$ being the deviance of the mean of the parameters. We therefore use $\overline{D(\boldsymbol{\theta})}$ as a measure of model fit and p_D as a measure of model complexity (Carlin and Louis 2009). Using these measures, we define DIC as the model fit plus the model complexity, or

$$DIC = \overline{D(\boldsymbol{\theta})} + p_D \quad (2.14)$$

$$= 2\overline{D(\boldsymbol{\theta})} - D(\bar{\boldsymbol{\theta}}). \quad (2.15)$$

While there is no intrinsic meaning of the DIC value, the difference between two models can give insight into model selection. As with AIC, BIC, and other information

criterion methods, the lower the DIC a model has, the better—with differences of 3 to 5 typically being the smallest differences that should be noted or considered (Carlin and Louis 2009).

DIC is very advantageous for a number of reasons. Besides its generality, Carlin and Louis (2009) note that DIC is easy to calculate during an MCMC run by simply monitoring $\boldsymbol{\theta}$ and $D(\boldsymbol{\theta})$. This often turns out to be much easier than computing Bayes' factors. Spiegelhalter et al. (2002) caution that p_D can be negative. While this would seem to be unsettling to have a negative value for an estimate of effective parameters, it can be used to check the agreement between priors and data. If $p_D < 0$, it typically suggests 'substantial conflict between the prior and data.' One drawback of the DIC is that it is not invariant to parameterization. This then requires the parameterization to be carefully considered beforehand. Overall, however, DIC should be strongly considered when comparing any models that have a hierarchical structure.

METHODS AND PROPOSAL

We now outline the three different models proposed in modeling missile reliability. We first go over the specifics of the models, along with details about how each will account for different features. Three different test cases are then laid out to test the models. Included in these test cases are the success criteria for each one. Finally, proposed steps to be used for model selection are outlined.

3.1 MODEL 1

We begin with Model 1, a base model, that will then be simplified for other models. Similar to the approach of Martz et al. (1988) and Martz and Waller (1990), we begin by dividing the system reliability up into subsystem reliabilities. These subsystem reliabilities will be the reliabilities of the components of the missile in the different phases of the missile. For simplicity, we assume the components are in series. This assumption not only simplifies the assumption but it also makes for a more conservative model. We then have

$$R_S = R_{1,1,1} * R_{2,1,1} * \dots * R_{C,P,1}, \quad (3.1)$$

where R_S is the system reliability and R_{ij1} is the reliability of the i^{th} component during the j^{th} phase under the 1st test conditions. C and P denote the number of total components and phases respectively.

We then turn to modeling the reliability of each of these subsystems. To do this, Bayesian logistic regression is used as suggested by Reese et al. (2005). We start by taking the logit transform, given in (2.1), of each subsystem reliability. We set this equal to the three explanatory variables: μ_{ij} , the underlying reliability for the

i^{th} component during the j^{th} phase; α_i , the age effect on the i^{th} component; and, λ_{ijk} , the effect of the k^{th} mode on the i^{th} component during the j^{th} phase. Note, λ_{ij1} is set to 0. This will allow the first modality to be the ‘base reliability’ or point of reference. We then have

$$\text{logit}(R_{ijk}) = \mu_{ij} + \alpha_i * \text{age} + \lambda_{ijk}. \quad (3.2)$$

We now present the likelihood that is used for the models. The likelihood shown in (3.3) is for the general case with n_c components and n_p phases. The general likelihood is defined as:

$$\begin{aligned} f(\underline{Z}, \underline{x} | \underline{R}) = & \\ & (R_{1,1} \dots R_{n_c,1})^{Z_1} \dots (R_{1,n_p} \dots R_{n_c,n_p})^{Z_{n_p}} \\ & (1 - R_{1,1})^{x_{1,1}} \dots (1 - R_{n_c,1})^{x_{n_c,1}} \dots \\ & (1 - R_{1,n_p})^{x_{1,n_p}} \dots (1 - R_{n_c,n_p})^{x_{n_c,n_p}}, \end{aligned} \quad (3.3)$$

where Z_j is the number of successes in the j^{th} phase and x_{ij} is the number of failures for i^{th} component during the j^{th} phase.

There are two important restrictions to keep in mind. First:

$$N = Z_1 + \sum_{i=1}^C x_{i1},$$

where N is the total number of tests and C is the number of components. Basically, the number of tests must be equal to the sum of the number of phase one successes and the sum of the failures in each component during phase one. Also, the number of successes in the previous phase must equal the sum of the number of successes in the next phase and the sum of the failures in each component during the next phase, or:

$$Z_{j-1} = Z_j + \sum_{i=1}^I x_{ij} \text{ for all } j = 2, \dots, P.$$

We now specify our priors. Since we have taken the logit transform, we are now only interested in estimating the parameters in (3.2), namely, μ_{ij} , α_i , and λ_{ijk} .

Because we believe each parameter is related to the parameters across components and phases ($\mu_{1,1}$ is related to $\mu_{2,1}$ and α_1 is related to α_3), we construct a hierarchical model. The priors are then:

$$\mu_{ij} \sim N(\omega_j, \sigma_\mu^2), \quad (3.4)$$

$$\alpha_i \sim N(\gamma, \sigma_\alpha^2), \quad (3.5)$$

$$\lambda_{ijk} \sim N(\eta_{jk}, \sigma_\lambda^2). \quad (3.6)$$

Given these priors, we must now specify priors for the following hyperparameters: ω_j , γ , and η_{jk} . These priors are as followed:

$$\omega_j \sim N(\omega^0, \sigma_\omega^2), \quad (3.7)$$

$$\gamma \sim N(0, \sigma_\gamma^2), \quad (3.8)$$

$$\eta_{jk} \sim N(\eta_{jk}^0, \sigma_\eta^2), \quad (3.9)$$

where ω^0 and η_{jk}^0 are constants, not hyperparameters. We now specify the priors for the following variance hyperparameters: σ_μ^2 , σ_α^2 , σ_λ^2 , σ_ω^2 , σ_γ^2 , and σ_η^2 .

$$\sigma_\mu^2 \sim IG(a_\mu, b_\mu), \quad (3.10)$$

$$\sigma_\alpha^2 \sim IG(a_\alpha, b_\alpha), \quad (3.11)$$

$$\sigma_\lambda^2 \sim IG(a_\lambda, b_\lambda), \quad (3.12)$$

$$\sigma_\omega^2 \sim IG(a_\omega, b_\omega), \quad (3.13)$$

$$\sigma_\gamma^2 \sim IG(a_\gamma, b_\gamma), \quad (3.14)$$

$$\sigma_\eta^2 \sim IG(a_\eta, b_\eta), \quad (3.15)$$

Now that priors and likelihood are outlined, Model 1 is completely specified.

3.2 MODEL 2

We now specify Model 2. This model is similar to Model 1 with the slight difference that we will only want to allow for one aging effect across all components. Instead of having different values for each α_i , there is only one common value, α . Thus, (3.2) is replaced with:

$$\text{logit}(R_{ijk}) = \mu_{ij} + \alpha * \text{age} + \lambda_{ijk}. \quad (3.16)$$

All the priors in Model 1 are the same except for now, α is no longer hierarchical and (3.5) is replaced by

$$\alpha \sim N(0, \sigma_\alpha^2). \quad (3.17)$$

Also, the priors (3.8) and (3.14) are no longer needed. As mentioned, all other priors are held the same.

3.3 MODEL 3

In Model 3, we estimate just the system reliability without dividing the reliability into subsystems as done in (3.1). We therefore are only interested in estimating R_S . Similar to (3.2) and (3.16), we will take the logit transform of R_S . Because we are now only looking at modeling one reliability there are no indices on μ or α . However, there will still be a modal index on λ because there are still different test conditions on each test. We therefore have:

$$\text{logit}(R_S) = \mu + \alpha * \text{age} + \lambda_k. \quad (3.18)$$

The priors for Model 3 are as followed:

$$\mu \sim N(\mu^0, \sigma_\mu^2), \quad (3.19)$$

$$\alpha \sim N(0, \sigma_\alpha^2), \quad (3.20)$$

$$\lambda_k \sim N(\eta_k, \sigma_\lambda^2), \quad (3.21)$$

where μ^0 is a constant. We now only have one parameter, λ_k , with a hierarchical structure. Therefore, our only hyperparameter prior is

$$\eta_k \sim N(\eta^0, \sigma_\eta^2), \quad (3.22)$$

where η^0 is a constant. The variance hyperparameters are the same as in Model 1, except priors (3.13) and (3.14) are no longer needed.

3.4 TEST CASES

In order to make sure the models work, we lay out four test cases, along with success criteria. In summary, we will have a null case with no aging, a constant aging case, a sudden onset degradation case, and a modality effect case. These will be outlined thoroughly below. Each illustrative cases will show the robustness of the models.

In all three test cases, C, the number of components, is fixed to 20; P, the number of phases, to 4; and M, the number of modalities, to 3. While these are not the exact numbers of the actual Cruise Missile, they will give sufficient evidence for success of the models. For each case there are 100 tests. The tests will consist of a uniform population of missiles aged 1-20 years and a uniform population of modalities for each test.

Case 1 - Null Case

The first case is a null case where no aging will occur. It is important to check to make sure we can model this simple case. In this null case, each missile will have a reliability of approximately 0.96, independent of missile age. We also assume each component, during each phase, has the same reliability. We then generate the data according to these probabilities. Ultimately, we want to make sure we estimate the underlying reliability correctly. Also, we want to make sure the slope of the aging effect is zero with high probability. This is important because we do not want to

overfit the model when there is no aging effect. Similarly, we want our modal effect to be zero with high probability. In order to check for these conditions, we outline the following success criteria.

Success Criteria:

1. Demonstration of Proper Mixing: Since the algorithm will implement the Metropolis-Hastings algorithm to generate observations from the posterior, we show there is a reasonable mixing rate. We want the acceptance rate for each parameter generated using the Metropolis-Hastings algorithm to be between 20-40%. If it is in this range, the criterion is satisfied.
2. Demonstration of Convergence: Since the test case data was generated such that the system reliability is approximately .96, this will be our target system reliability. We therefore expect the posterior median reliability for the 100-test data set to be in the interval (0.925, 0.975).
3. Demonstration of No Aging: Since there is no aging effect in the data, it is important that the aging effect, α_i , is zero with high probability for all i . To check this we consider

$$P_{\alpha_i} = \min(\Pr(\alpha_i > 0|data), \Pr(\alpha_i < 0|data)).$$

If $P_{\alpha_i} < 0.10$ for any component i , then we conclude that there is high probability that there is an aging effect, and the test has *failed*.

4. Demonstration of No Modality Effect: Similar to the aging effect, since there is no modality effect in the data, we will want to demonstrate the model does not try to estimate a modality effect. To do this we examine

$$P_{\lambda_{ijk}} = \min(\Pr(\lambda_{ijk} > 0|data), \Pr(\lambda_{ijk} < 0|data)).$$

Again, if $P_{\lambda_{ijk}} < 0.10$ for any i, j, k , then we conclude that there is high probability that there is a modal effect, and the test has *failed*.

Case 2 - Constant Aging Case

The next case is for a constant aging effect. Because it is difficult to explain and interpret the aging results in terms of system reliability, we will deviate from the null case and talk about the constant aging case in terms of μ_{ij}, α_i , etc. Recall the logit transform explained in Section 2.4,

$$\log\left(\frac{R_{ij}}{1 - R_{ij}}\right) = \text{logit}(R_{ij}) = \mu_{ij} + \alpha_i * \text{age} + \lambda_{ijk}.$$

We set $\mu_{ij} = 7.6 \forall i, j$, which means $R_{ij} \approx 0.9995$ and $R_S \approx 0.96$ at time 0. An aging effect is introduced by letting $\alpha_2 = -0.05$ and $\alpha_5 = -0.2$. For $i \neq 2, 5$, α_i is set to zero. For Model 1 and Model 2, it is important to be able to recognize when there is an aging effect on certain components and not on others. Also, similarly to the null test case, we want our model to correctly identify the aging effect without altering or incorrectly identifying the underlying reliability of each component (R_{ij}).

Success Criteria:

1. Demonstration of Proper Mixing: As in Case 1, we want a mixing rate of 20-40% for all parameters that are generated from the Metropolis-Hasting algorithm. This ensures proper mixing and a sufficient exploration of the parameter space.
2. Demonstration of Convergence: Assuming criterion 1 is met, we want the model to approximate the true values for μ_{ij} , which is 7.6. Therefore, we want the median of the posterior for each μ_{ij} to be in the range (6.5, 7.9), or μ_{ij} 's are such that the induced system reliability, R_S , is in the range (0.925, 0.975). If this holds then the test meets the criterion.
3. Demonstration of Aging: Since there is no aging effect on any components except for 2 and 5, we first want to make sure there is no evidence of aging on the other components. As mentioned above, we will examine

$$P_{\alpha_i} = \min(\Pr(\alpha_i > 0 | \text{data}), \Pr(\alpha_i < 0 | \text{data})).$$

If, for any $i \neq 2, 5$, $P_{\alpha_i} < 0.1$, then we have evidence of aging on these components and the test *fails*. Alternatively, if $P_{\alpha_i} > 0.1$ for $i = 2, 5$, then there is not sufficient evidence of aging when there should be and the test *fails*.

4. Demonstration of No Modality Effect: Since there is again no modality effect, we make sure the model does not estimate an effect. To check this we repeat the test presented in the null case test.

Case 3 - Sudden Degradation Onset Case

In the third test case, we look at the case of sudden onset aging. This occurs when reliability of a missile is constant for the beginning of the missile's life but begins to decrease at a certain age. Currently, the model is built to only handle cases where the aging is constant through time. Since this is a common type of aging, we would like to make sure our model can give a somewhat reasonable estimate if this were to happen.

For this test we again let $\mu_{ij} = 7.6$ for all i, j . We then add an aging effect on component 5 and let $\alpha_5 = -.2$. However, this aging effect will start at age 11. Therefore, we get

$$\text{logit}(R_{5jk}) = \mu_{5jk} + \alpha_5 * (\text{age} - 10).$$

In this test case, we just consider a sudden onset aging effect, and therefore, do not add a modality effect.

Success Criteria:

1. Demonstration of Proper Mixing: As described in the other tests, we will be using the Metropolis-Hastings algorithm to generate draws from our posterior. Therefore, we make sure there is a proper mixing rate for all parameters generated using the Metropolis-Hastings algorithm. Again, the criterion for the acceptance rate is the range 20–40%.

2. Demonstration of Convergence: Assuming criterion 1 is met, we want the model to approximate the true values for μ_{ij} , which is 7.6. Therefore, the median of the posterior for each μ_{ij} should be in the range (6.5, 8.0), or μ_{ij} 's are such that the induced system reliability, R_S , is in the range (0.925, 0.975). If this holds than the test meets the criterion.
3. Demonstration of Aging: If criteria one and two are met, we show the aging effect. Since there should be no aging effect on any components except for component 5, we first show that α_i is zero with high probability for all $i \neq 5$. We consider P_{α_i} as described above. If $P_{\alpha_i} > 0.1$ for all $i \neq 5$, it passes the initial test and we look at P_{α_5} . Since there is not a constant aging effect, we would just like to see that $\alpha_5 \neq 0$ with high probability. Hence, if $Pr(\alpha_5 \geq 0) > 0.1$ then the test fails. Otherwise, the test succeeds.
4. Demonstration of No Modality Effect: Finally, since we have not added a modality effect, we consider $P_{\lambda_{ijk}}$, as it is defined above. If $P_{\lambda_{ijk}} < 0.1$, then the test fails.

Case 4 - Modal Effect

In the final test case we would like to add a modality effect. Again, let $\mu_{ij} = 7.6 \forall i, j$. In this case we have no aging effect. However, we do have a modality effect. Mode 2 has a positive effect on reliability on component 1 with a value, $\lambda_{1j2} = 1.5$ for $j=(1,2,3,4)$. Mode 3, on the other hand, has a negative effect on reliability for component 2 and phase 3 with a value, $\lambda_{2,3,3} = -2$. It is important to find these modal effects because we do not want to overestimate or underestimate the underlying reliability based on what type of test is run. This allows us to account for the differences in these test types.

Success Criteria:

1. Demonstration of Proper Mixing: Again, the criterion for the acceptance rate is the range 20–40%.
2. Demonstration of Convergence: Assuming criterion 1 is met, the model should approximate the true values for μ_{ij} , which is 7.6. Therefore, we want the median of the posterior for each μ_{ij} to be in the range (6.5, 8.0), or μ_{ij} 's are such that the induced system reliability, R_S , is in the range (0.925, 0.975). If this holds than the test meets the criterion.
3. Demonstration of No Aging: Since no aging effect has been added for this test, α_i should be zero with high probability for all i . To test this we consider P_{α_i} where,

$$P_{\alpha_i} = \min(\Pr(\alpha_i > 0|data), \Pr(\alpha_i < 0|data)).$$

If $P_{\alpha_i} < 0.1$ for any i , then the test *fails*.

4. Demonstration of Modal Effect: The target for the modal effect is $\lambda_{1j2} = 1.5$ for $j=1,2,3,4$, $\lambda_{2,3,3} = -2$, and $\lambda_{ijk} = 0$ for all other i, j, k . So, the median of $\lambda_{1,j,2}$ should be in the range (1.25, 1.75) for all j , the median of $\lambda_{2,3,3}$ in the range (-2.25, -1.75), and λ_{ijk} for all other i, j, k , to be zero with high probability. Again, we consider $P_{\lambda_{ijk}} > 0.10$. If this holds for all other i, j, k , then the criterion is met.

3.5 MODEL SELECTION

After fitting each of the models to the four different data sets we use model selection to select the preferred model to use. We start by checking the test case success criteria. Assuming all criteria are satisfied we proceed by comparing AIC, BIC, and DIC. If not all success criteria are met, we find the model that succeeds in the most test criteria and proceed with comparing the different information criterions. Since these models are hierarchical, we will use the Deviance Information Criterion that

was set out in Section 2.8 as the most important of these measures. This is reported in Section 4.4.

EVALUATION OF THE MODELS

4.1 INTRODUCTION

As discussed in Chapter 3, three different models are proposed to account for the reliability of the missile. We start with Model 1, the full model, where there is a unique μ_{ij} for each component, i , and phase, j , combination. Also, Model 1 has a unique aging parameter, α_i , for each component. Additionally, each component, phase, and modality combination has a unique modality effect, denoted λ_{ijk} . Model 2 is the same as Model 1 except for now all components have the same aging parameter. Model 3 has only μ , α , and λ parameters at the system level. Essentially, we summarize all the μ_{ij} into a single μ_S parameter. Similarly, we do the same for α_i and λ_{ijk} . For more details on these models refer to Sections 3.1, 3.2, and 3.3.

Section 4.2 discusses the details of the MCMC algorithm used in parameter estimation. Section 4.3 discusses the success or failure of the test cases outlined in Section 3.4. Lastly, Section 4.4 summarizes Section 4.3 and discusses final model comparisons using DIC which was outlined in Section 2.8.

4.2 MCMC ALGORITHM

In order to generate draws from the posterior, and for computational efficiency, the MCMC algorithm was coded using MATLAB. Throughout the algorithm, μ_{ij} , α_i , and λ_{ijk} for all i, j , and k were updated using a Metropolis-Hastings step. All other variance parameters ($\sigma_\mu^2, \sigma_\alpha^2, \sigma_\eta^2$, etc.) and hyperparameters (η_{jk}, ω_j , and γ) that were outlined in Section 3.1 are updated using a Gibbs sampling step. This code can be seen in Appendix A. This code took significant computation time to run 200,000

iterations for each model and data set. Model 1 and Model 2 each required approximately 7 days per data set. Model 3 took about 2 hours of computation time per data set to run. This required $(7 \text{ days} \times 2 \text{ models} \times 4 \text{ data sets}) + (2 \text{ hours} \times 4 \text{ data sets}) \approx 56.33$ days of computation time.

On top of the time required to run the actual algorithms, it is also worth noting the difficulty and amount of time consumed in dealing with the tuning parameters. When using any Metropolis-Hastings steps, one must choose the standard deviation for the candidate distribution such that the parameter space is properly ‘explored.’ If too large of a standard deviation is chosen, not enough draws from the posterior will be accepted. This causes the posterior draws to not cover the entire space of the actual posterior. If too small of a standard deviation is chosen, the draws could potentially get hung up at a local maxima and, again, not cover the true entire posterior space. The first success criterion for each data set ensures a proper standard deviation was chosen. This standard deviation was chosen manually, using trial and error, for each parameter (i.e. the standard deviation for the candidate values for all μ_{ij} ’s is the same but different than the standard deviation for the candidate values for all the α_i ’s).

4.3 SUCCESS CRITERIA

Recall the success criteria outlined in Section 3.4. Before proceeding to compare the different models using DIC, we would like to check the models to see how successful they are in identifying different phenomena in the data sets provided. Each of the four data sets provides a different test case as described in Section 3.4. We check different success criteria for all three models before proceeding to the next data set.

Data Set 1

This data set is the null case (i.e. no aging and no modal effect). This is essentially a base case. We would like to see the following:

1. Proper mixing (20% - 30% acceptance rate) for each parameter.
2. Demonstration of convergence to the ‘true’ values.
3. Demonstration of no aging.
4. Demonstration of no modal effect.

MODEL 1

1. The mixing rates of the different parameters are first checked. Table 4.1 shows the different mixing rates. All mixing rates are in the 20-40% range, and, therefore, the criterion is satisfied.

Parameter	Mixing Rate
μ	30.91%
α	29.80%
λ	39.23%

Table 4.1: Mixing rates for parameters of Model 1 for null case. All are within the acceptable 20-40% range.

2. Checking to make sure the model is converging on the correct underlying reliability, we see that the median of μ_{ij} is contained in the range (6.233,7.929) for all i and j . Most, but not all, of these are in the acceptable range; however, when we convert these back to component reliabilities and find the induced system reliability, R_S , we find that $R_S \approx .962$. Since this is in the range outlined in Section 3.4, we conclude the test case has succeeded.

3. To ensure there is no aging effect, we consider P_{α_i} , as defined in Section 3.4. We would like to see $P_{\alpha_i} > 0.1$ to indicate that there is no aging effect. However, using this model we see that $P_{\alpha_i} < 0.1$ for 17 of the 20 components. We therefore conclude that there is an aging effect, and the test case has failed for this criterion. Looking at Figure 4.1 we see that there is a significant positive aging effect. This is a problem that will be addressed in Section 5.2.

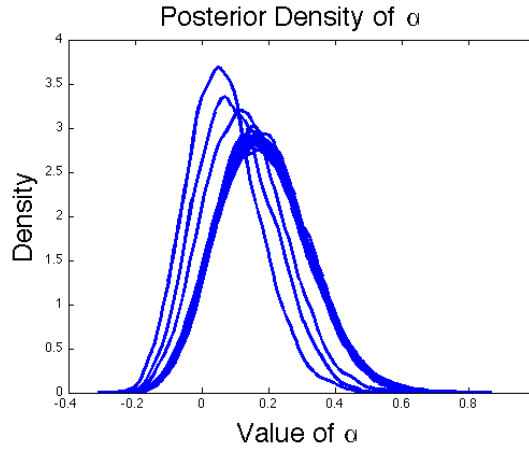


Figure 4.1: Posterior probability density of aging parameter for null case with no aging shows there is a positive aging effect using Model 1.

4. Finally, we look to see if there is a modality effect. We consider $P_{\lambda_{ijk}}$, for all i, j , and k . Since $P_{\lambda_{ijk}} \in (.3326, .4999)$ for all i, j , and k , we say there is no significant modality effect. Therefore, Model 1 passes criterion four for Data Set 1.

MODEL 2

1. The mixing rates of the different parameters are first checked. Table 4.2 shows the different mixing rates. All mixing rates are in the 20-40% range, and, therefore, the criterion is satisfied.

Parameter	Mixing Rate
μ	33.23%
α	23.23%
λ	40.12%

Table 4.2: Mixing rates for parameters of Model 2 for null case. All mixing rates are in the desired 20-40% range

2. Checking to make sure the model is converging on the correct underlying reliability, we see that the median of μ_{ij} is contained in the range (6.06,7.92) for all i and j . Converting each μ_{ij} and solving for R_S , we find $R_S \approx .961$. The test criterion is therefore met.
3. To check for the effect of aging, we consider P_α . Using Model 2, $P_\alpha = .1441$, and we conclude there is no significant aging effect. Figure 4.2 shows the density of α . Recall, in Model 2 there is only one aging parameter that is shared for every component. Therefore, only one density is shown.

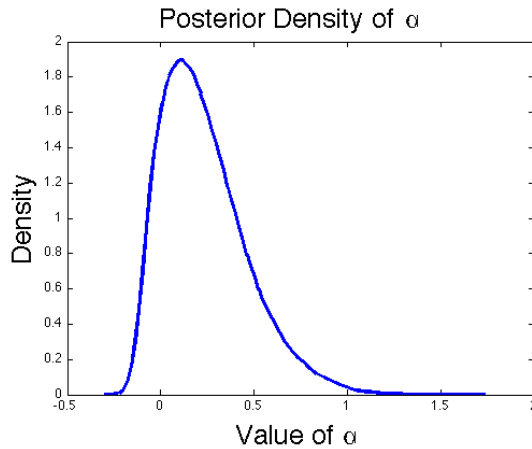


Figure 4.2: Posterior probability density of aging parameter for null case with no aging. Aging effect is 0 with high probability.

4. Again, we demonstrate that there is no modality effect, this time for Model 2. Since $P_{\lambda_{ijk}} \in (0.3386, 0.4999)$ for all i, j , and k , we conclude there is no modal effect. Therefore, Model 2 succeeds in passing this criterion for Data Set 1.

MODEL 3 As described before, this model is a simplified version where the underlying reliability and different effects (age and different modalities) are estimated at only the system level.

1. The mixing rates of the different parameters are first checked. Table 4.3 shows the different mixing rates. All mixing rates are in 20-40% range, and, therefore, the criterion is satisfied.

Parameter	Mixing Rate
μ	36.84%
α	37.67%
λ	35.99%

Table 4.3: All mixing rates for parameters of Model 3 in the null case are in the acceptable 20-40% range.

2. We now consider the distribution for μ , the underlying reliability. There is only one μ parameter for the entire system. Therefore, the value of μ will not be close to the generated value for each component of 7.6. As mentioned earlier, a value of $\mu_{ij} = 7.6$ corresponds to $R_{ij} \approx 0.9995$. This implies that the reliability of the system, $R_S \approx 0.95$, or $\mu_S \approx 2.944$. The mean of the posterior distribution of $\mu_S = 3.668$, or $R_S \approx 0.975$. This is an acceptable number for R_S , and therefore the criterion is met. Figure 4.3 shows the density of μ_S , the underlying system reliability, as well as a trace plot of μ from the MCMC algorithm.

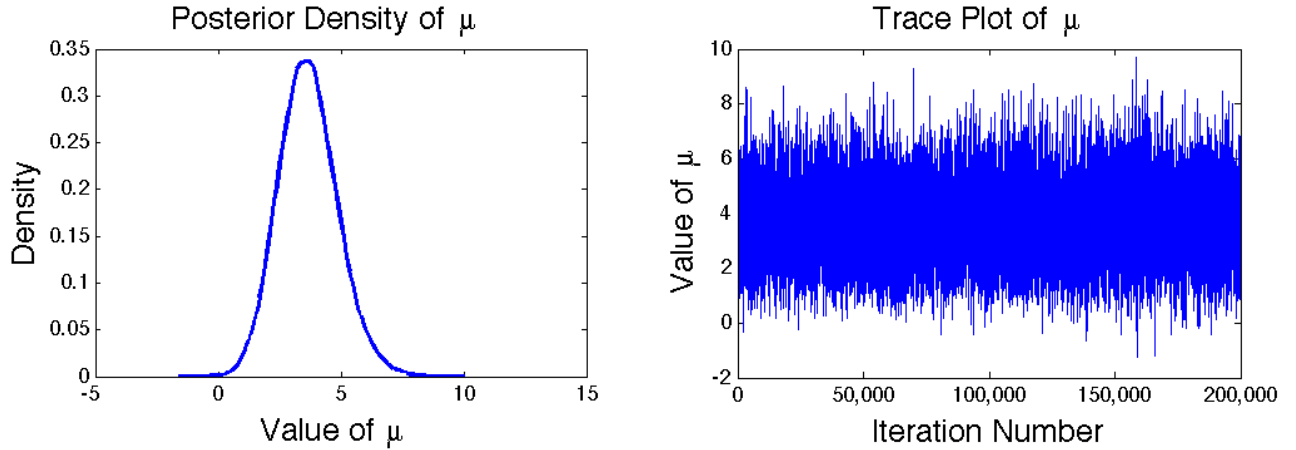


Figure 4.3: Posterior probability density of μ_S for Model 3 in the null case on the left. Trace plot of μ_S (right) shows the algorithm has converged.

3. To check for the effect of aging, we consider P_α . For this data and model, $P_\alpha = 0.2642$, and we conclude there is no significant aging effect. Figure 4.4 shows the density of α , the aging effect.

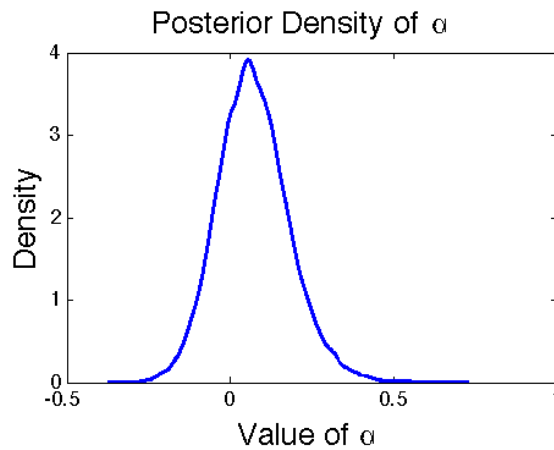


Figure 4.4: Posterior probability density of aging effect for the null case with no aging using Model 3 shows the aging effect is 0 with high probability.

4. Finally, we demonstrate that there is no modality effect. Since $P_{\lambda_2} = .03684$ and $P_{\lambda_3} = 0.2304$ (recall λ_1 is constrained to 0), we conclude there is no significant

modality effect and Model 3 passes this criterion for Data Set 1. Figure 4.5 shows the density of λ_2 and λ_3 .

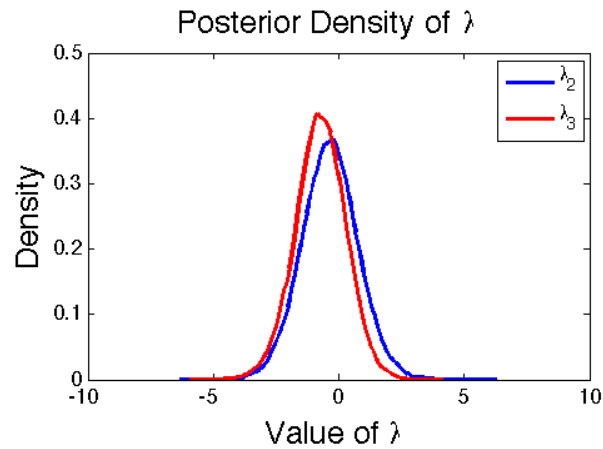


Figure 4.5: Posterior probability density of modality effect using Model 3 for null case shows no significant effect.

Data Set 2

This next data set is the constant aging case. As described in Section 3.4 there is a negative effect on certain components. Therefore, we would like to see the following:

1. Proper mixing (20% - 30% acceptance rate) for each parameter.
2. Demonstration of Convergence to the true values.
3. Demonstration of aging.
4. Demonstration of no modality effect.

Again, the importance of Data Set 2 is to see the effectiveness of each model at identifying aging effects. All three models successful complete criteria one, two, and four. We, therefore, concentrate on criterion three.

MODEL 1 We begin with Model 1. With this model, each component has a unique α parameter. As mentioned before, we would like to see an aging effect on components 2 and 5. On all other components, we would like to not see a significant aging effect. For this model $P_{\alpha_2} = 0.2089$, $P_{\alpha_5} = 0.0148$, and $P_{\alpha_i} \in (0.1981, 0.4493)$ for all other i . We, therefore, see a significant aging effect on component 5 and no significant aging effect on the other components. While the aging effect on component 5 is desired, we would have also liked to see an aging effect on component 2. Figure 4.6 displays the posterior densities for α parameters for each component. α_2 and α_5 are highlighted as noted in the legend.

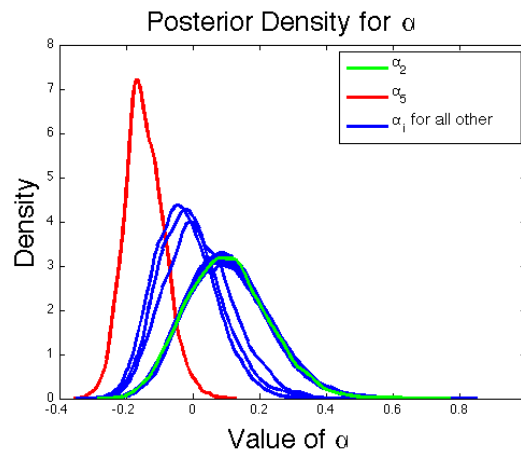


Figure 4.6: Posterior probability density of aging effect for Model 1 in the constant aging case. Component 5 has a significant degradation due to aging.

MODEL 2 To check for the effect of aging, we again consider P_α . Using Model 2, $P_\alpha = 0.3752$, and we conclude there is no significant aging effect. The criterion, therefore, fails for this model on Data Set 2. Figure 4.7 shows the density of α . Recall, in Model 2 there is only one aging parameter that is shared for every component. Therefore, only one density is shown. It is interesting to note that the density for α for Data Set 2 is similar to that of α in Data Set 1, except it is shifted more negative. This shift possibly suggests the model is slightly picking up the aging effect.

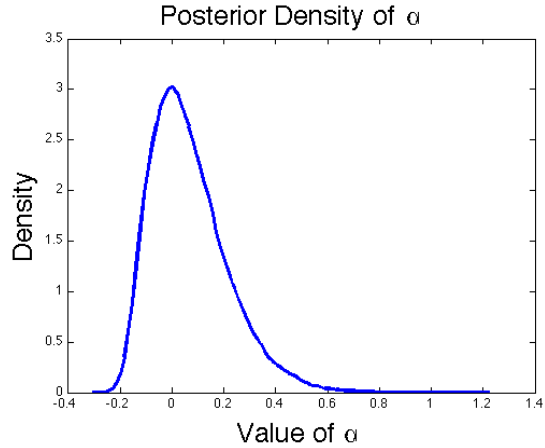


Figure 4.7: Posterior probability density of aging effect for Model 2 in the constant aging effect. Model 2 found no significant aging effect.

MODEL 3 Using Model 3, there is only one aging parameter that is shared for the entire system. Since we induced the aging effect by fixing some of the components to have negative values for α , we would like to make sure there is a negative aging effect at the system level. $P_\alpha = 0.0137$ indicates there is an aging effect. Looking at Figure 4.8, the posterior density of α_S , we see that it indeed is negative and the criterion is met.

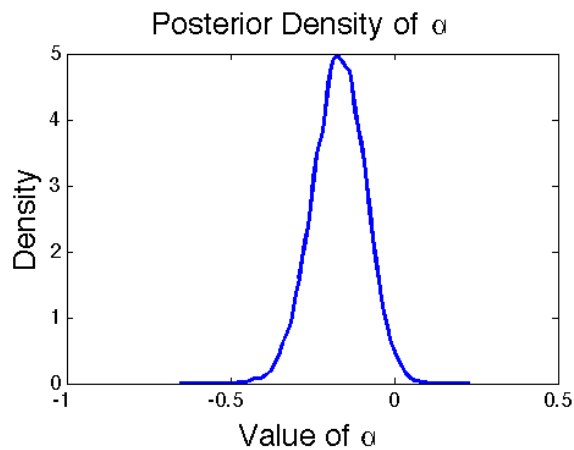


Figure 4.8: Posterior probability density of aging effect for Model 3 in the constant aging effect. A significant degradation was found using Model 3.

Data Set 3

Data Set 3 is the sudden onset aging case. Here, the aging does not start until after age 10. While none of the models are built to specifically handle this type of aging, we check to see how each one does for this phenomena. We want to see the following:

1. Proper mixing (20% - 30% acceptance rate) for each parameter.
2. Demonstration of convergence to the true values.
3. Demonstration of aging.
4. Demonstration of no modality effect.

Specifically, Data Set 3 is intended to see how well the models are at picking up an aging effect when the aging degradation has a sudden onset. Similarly to Data Set 2, all three models successfully complete criteria one, two, and four, and, therefore, concentration should be focused on criterion three. Recall, component 5 had a sudden onset aging effect after age 10. We therefore would like to see some aging effect represented in each model.

MODEL 1 Starting with Model 1, we again investigate P_{α_i} for each component. We see $P_{\alpha_5} = 0.4189$ and $P_{\alpha_i} \in (0.1628, 0.4959)$. Therefore, we do not see a significant aging effect for any of the components and this test case has failed. Figure 4.9 shows the posterior distributions for each of the α_i parameters.

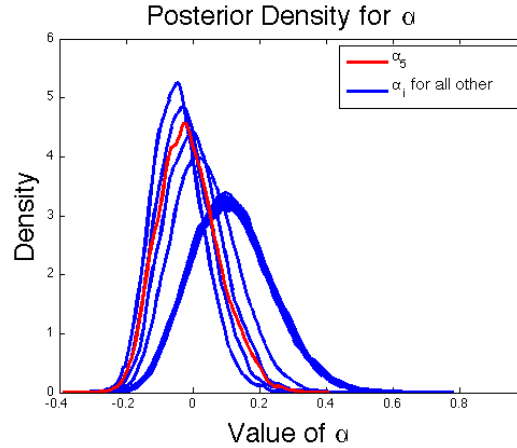


Figure 4.9: Posterior probability density of aging effect for Model 1 in the sudden onset aging case. There was no significant aging effect using Model 1.

MODEL 2 To check for the effect of aging, we again consider P_α . Using Model 2, $P_\alpha = 0.2813$, and we conclude there is no significant aging effect. The criterion therefore fails for this model on Data Set 2. Figure 4.10 shows the density of α . Recall, in Model 2 there is only one aging parameter that is shared for every component. Therefore, only one density is shown. Similar to Data Set 1 and 2, it is interesting to note the density for α . Again, the shape is similar to the density of α for Data Set 1 and 2; however, it's shifted more negative than the null case (Data Set 1) and slightly more positive than the aging case (Data Set 2). This again, possibly suggests Model 2 is slightly picking up the aging effect.

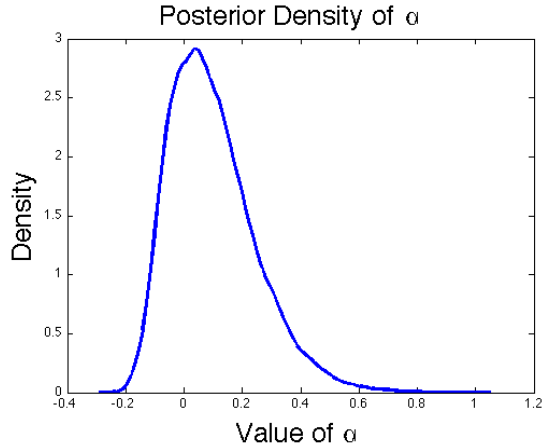


Figure 4.10: Posterior probability density of aging effect for Model 2 in the sudden onset aging case shows no significant effect.

MODEL 3 Using Model 3, there is only one aging parameter that is shared for the entire system. Again, we want to make sure there is a negative aging effect at the system level. $P_\alpha = 0.3066$ indicates there is not a significant aging effect. Looking at Figure 4.11, the posterior density of α_S , we see that while the aging effect appears to be negative, it is not significant. Therefore, the criterion is not met for the sudden onset degradation.

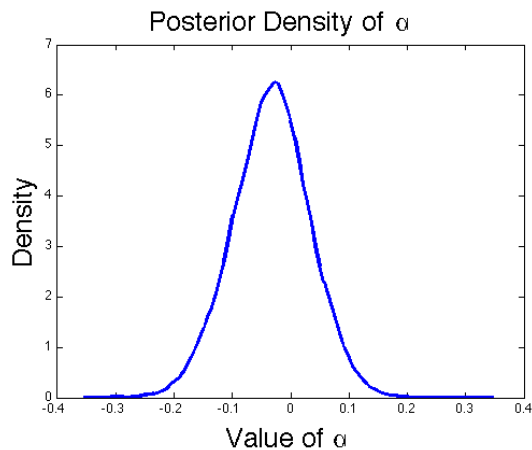


Figure 4.11: Posterior probability density of aging for Model 3 in the sudden onset aging case shows no significant aging effect.

Data Set 4

Finally, Data Set 4 is the test case with different modality effects in the data set. The specifics of the effects of each modality are given in Section 3.4. We would like to see the following:

1. Proper mixing (20% - 30% acceptance rate) for each parameter.
2. Demonstration of convergence to the true values.
3. Demonstration of no aging.
4. Demonstration of modality effect.

The main focus of Data Set 4 is the demonstration of the modality effect. All three models successfully complete criteria one, two, and three. Therefore, the results presented below are only for criterion four.

MODEL 1 Using Model 1, we see $P_{\lambda_{ijk}} \in (0.3323, 0.4999)$ for all i, j , and k . This implies there is no modal effect. This goes against what we would like to find. Therefore, we conclude Model 1 has failed the criterion and has failed this test case.

MODEL 2 Similarly, for Model 2 we find $P_{\lambda_{ijk}} \in (0.3338, 0.4998)$ for all i, j , and k . This once again fails to identify the modality effect. Model 2, then, fails to pass the test case for the different modality effect.

MODEL 3 Using Model 3, we see that $P_{\lambda_2} = 0.4522$ and $P_{\lambda_3} = 0.4681$. Again, we conclude that there is no modality effect and, therefore, implies that Model 3 also fails to succeed the criteria for this test case. Figure 4.12 displays the posterior densities of λ_2 and λ_3 .

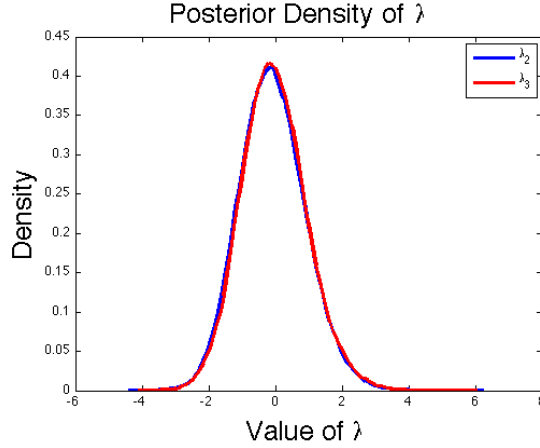


Figure 4.12: Posterior probability density of modality effect for Model 3 in the modality effect case shows no significant effect.

4.4 MODEL COMPARISON AND DISCUSSION

We begin by presenting a summary of Section 4.3. Table 4.4 summarizes the success of each model for the different data sets. X's indicate successful completion of success criteria for the test cases. Blank squares indicate that the specified model failed the indicated criterion for the test case.

Success Criteria	Model 1				Model 2				Model 3			
	1	2	3	4	1	2	3	4	1	2	3	4
Data Set 1 (Null Case)	X	X		X	X	X	X	X	X	X	X	X
Data Set 2 (Aging Case)	X	X	X	X	X	X		X	X	X	X	X
Data Set 3 (Sudden Aging Case)	X	X		X	X	X		X	X	X		X
Data Set 4 (Modality Case)	X	X	X		X	X	X		X	X	X	

Table 4.4: Model 3 outperformed other models with respect to test case criteria.

We now discuss each data set, how the models performed, and the DIC for each model in each data set. For a definition and explanation of DIC, see the subsection

entitled DIC in Section 2.8. For each data set, a table lists the DIC values and the Pd, or effective number of parameters, for each model.

Data Set 1

As we can see in Table 4.4, Models 2 and 3 do a better job of identifying certain characteristics of Data Set 1. Model 1 did a poor job of correctly identifying the aging effect. Using Table 4.5, we see that not only does Model 3 successfully pass each criterion, but also has the lowest DIC for Data Set 1.

	DIC	Pd
Model 1	56.1	6.38
Model 2	64.4	7.64
Model 3	32.0	2.75

Table 4.5: Model 3 has significantly lower DIC values for null case.

Data Set 2

For Data Set 2, Table 4.4 indicates that Model 1 and 3 successfully complete each of the test criterion. While Model 2 appears to somewhat identify the aging effect, it is not significant, and therefore, fails the criterion. As in Data Set 1, not only does Model 3 complete all the test criteria, but it also has the lowest Pd and DIC values, as shown in Table4.6.

	DIC	Pd
Model 1	92.1	10.30
Model 2	118.2	13.20
Model 3	44.7	3.34

Table 4.6: Model 3 has significantly lower DIC values for constant aging case.

Data Set 3

While none of the three models successfully identified the aging effect in Data Set 3, we still compare the DIC. Once again, Model 3 has the lowest DIC, as can be seen in Table 4.7. It is important to note that while none of the models found a significant aging effect, it does appear that each of the models found the aging effect to be more negative (the desired effect) than the null case.

	DIC	Pd
Model 1	128.2	11.44
Model 2	145.0	13.80
Model 3	61.8	3.60

Table 4.7: Model 3 has significantly lower DIC values for sudden onset aging case.

Data Set 4

Finally, we look at Data Set 4. It appeared that none of the models did a satisfactory job at identifying differences in modality. They all failed the test case criteria. However, looking at Table 4.8, we once again see that Model 3 has the lowest DIC of the three models.

	DIC	Pd
Model 1	84.9	7.60
Model 2	93.9	9.06
Model 3	45.6	3.13

Table 4.8: Model 3 has significantly lower DIC values for modality effect case.

CONCLUSIONS**5.1 MODEL SELECTION**

We now choose which model of the three we propose to use. After looking at the results and discussions in Section 4.4, it is easy to conclude that Model 3 is the most efficient of the three models. Not only does Model 3 have the lowest DIC in every test case, but it also successfully passed the most test criteria, and, maybe most important, it was the quickest to run simulations. As discussed in Section 4.2, Model 1 and Model 2 each take approximately 7 days of computation time to run. Alternatively, Model 3 takes approximately $1\frac{1}{2}$ to 2 hours. This improvement by a factor of more than 80 is extremely significant and makes Model 3 much more reasonable to run and use. The only downside of Model 3 is that it does not give any information about the specific reliability of the different components. If this information is desired, then Model 1 appears to be a more appropriate model.

5.2 FURTHER WORK

While we have proposed three very reasonable models, there are few areas of work that could be done. This work could improve the models and would shed additional insight on the models' efficiency. To improve the models, the handling of the aging effect needs to be modified, the modality effect needs to be adjusted, and the effect of sample size needs to be analyzed and addressed.

Distribution for Aging Effect

In the models proposed here, all the aging parameters were modeled as coming from a normal distribution. This allows positive and negative values for the α , aging effect. Often in these models, unless the aging effect was very significant and obvious, the aging effect would ‘drift’ so that it was a positive value and the underlying reliability, μ , would decrease. This is a very undesirable effect. To overcome this problem, we propose α be restricted to only take on negative values. This is reasonable because it seems extremely unlikely that something such as a missile, or missile component, would ever become more reliable over time. Therefore, only negative values for α seem reasonable. To do this, we propose the negative of the aging effect come from a gamma distribution, or $(-\alpha) \sim \text{gamma}(a_\alpha, b_\alpha)$. Using this proposal, considerations would have to be taken to decide what is significant. Using P_α , as defined in Section 3.4 and used in Section 4.3, would no longer make sense because the aging parameter, α , is now always less than 0.

Fixing modality

As shown in the discussion of Data Set 4 in Sections 4.3 and 4.4, the current models do not do a sufficient job in identifying different effects due to modality. This result is somewhat puzzling. While steps were taken to make sure the code was written and implemented correctly, it is possible that the correct hierarchy is not used for the modality effects. Further work is suggested to look into where the problems lie and correcting the modality effect.

Sample Size

Finally, this study only used one sample size for all four data sets, namely 100 tests for each one. While this seems like a reasonable amount of tests to use, some work

could be done to see the efficiency of each model as a function of sample size. Because Model 1 and Model 2 had so many parameters, 279 and 260 respectively, and Model 3 had so few parameters, 13 parameters, it is possible that Model 3 did better because it had to estimate fewer parameters. It would be interesting to see how Model 1 and Model 2 perform with additional test outcomes in each data set. This, however, would increase the computation time needed for each one, which is already substantial.

BIBLIOGRAPHY

- Akaike, H. (1974), “A New Look at the Statistical Model Identification,” *IEEE Transactions on Automatic Control*, 19, 716–723.
- Burnham, K. P., and Anderson, D. R. (2002), *Model Selection and Multimodel Inference: A Practical Information-Theoretic Approach* (2nd ed.), New York, NY: Springer.
- Carlin, B. P., and Louis, T. A. (2009), *Bayesian Methods for Data Analysis* (3rd ed.), New York, NY: Chapman & Hall.
- Casella, G., and George, E. I. (1992), “Explaining the Gibbs Sampler,” *The American Statistician*, 46, 167–174.
- Chernoff, H., and Lehmann, E. L. (1954), “The Use of Maximum Likelihood Estimates in χ^2 Tests for Goodness of Fit,” *The Annals of Mathematical Statistics*, 25, 579–586.
- Chib, S., and Greenberg, E. (1995), “Understanding the Metropolis-Hastings Algorithm,” *The American Statistician*, 49, 327–335.
- Cramér, H. (1946), *Mathematical Methods of Statistics*, Princeton, NJ: Princeton University Press.
- DiCiccio, T. J., Kass, R. E., Raftery, A., and Wasserman, L. (1997), “Computing Bayes Factors by Combining Simulation and Asymptotic Approximations,” *Journal of the American Statistical Association*, 92, 902–915.

- Gelfand, A. E., and Smith, A. F. M. (1990), “Sampling Based Approaches to Calculating Marginal Densities,” *Journal of the American Statistical Association*, 85, 398–409.
- Hamada, M. S., Wilson, A. G., Reese, C. S., and Martz, H. F. (2008), *Bayesian Reliability*, New York, NY: Springer.
- Hastie, T., Tibshirani, R., and Friedman, J. (2009), *The Elements of Statistical Learning* (2nd ed.), New York, NY: Springer.
- Hastings, W. K. (1970), “Monte Carlo sampling methods using Markov chains and their applications,” *Biometrika*, 57, 97–109.
- Johnson, V. E. (2004), “A Bayesian χ^2 Test for Goodness-of-Fit,” *The Annals of Statistics*, 32, 2361–2384.
- (2007), “Bayesian Model Assessment Using Pivotal Quantities,” *Bayesian Analysis*, 2, 719–734.
- Johnson, V. E., Graves, T., Hamada, M., and Reese, C. (2003), “A Hierarchical Model for Estimating the Reliability of Complex Systems,” *Bayesian Statistics*, 7, 199–213.
- Martz, H., and Waller, R. (1990), “Bayesian Reliability Analysis of Complex Series/Parallel Systems of Binomial Subsystems and Components,” *Technometrics*, 32, 407–416.
- Martz, H., Waller, R., and Fickas, E. (1988), “Bayesian Reliability Analysis of Series Systems of Binomial Subsystems and Components,” *Technometrics*, 30, 143–154.
- Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953), “Equations of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, 21, 1087–1092.

- Pearson, K. (1900), “On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to have arisen from random sampling,” *Philosophy Magazine*, 50, 157–172.
- Reese, C. S., Hamada, M., and Robinson, D. (2005), “Assessing System Reliability by Combining Multilevel Data from Different Test Modalities,” *Quality Technology & Quantitative Management*, 2, 177–188.
- Robinson, D. G., and Chang, I. A. (2003), “An Application of Bayesian Methods for Combining Data from Different Test Modalities,” *Sandia National Laboratories Sandia Report SAND*, 2002–3953.
- Schwarz, G. (1978), “Estimating the Dimension of a Model,” *The Annals of Statistics*, 6, 461–464.
- Spiegelhalter, D. J., Best, N. G., Carlin, B. P., and van der Linde, A. (2002), “Bayesian Measure of Model Complexity and Fit,” *Journal of the Royal Statistical Society B*, 64, 583–639.

APPENDICES

MATLAB CODE**A.1 MAIN PROGRAM**

```
function [] =MainProgram(testdata,modelnum,iterationnum)

burnin = floor(iterationnum/10);

iterationnum=iterationnum+burnin;

if modelnum==1
    SystemOnly=0;
    DiffAge=1;
elseif modelnum==2
    SystemOnly=0;
    DiffAge=0;
else
    SystemOnly=1;
    DiffAge=0;
end

YesAge=1;

%Start timer to get Elapsed Time
timeElapsed = tic;

%Format
format long

directory=cd;
```

```

filename = sprintf('%s/Data/%s.csv', directory, testdata);

if(SystemOnly==1)
    priorfilename = sprintf('%s/Data/PriorData2.csv', directory);
else
    priorfilename = sprintf('%s/Data/PriorData.csv', directory);
end

% number of iterations we want in MCMC

% if ACMorALCM = 1 then ALCM otherwise if ACMorALCM = 0 then ACM
ACMorALCM = 1;

% AgePartitions is the number of points on the Age axis that you want to
% explicitly
% compute the reliability. This is important when we want to summarize the
% reliability in graphs.
AgePartitions = 10;

% age bins is the number of bins on the age axis we want to use to compute
% the Bayesian Chi-Square statistic
nagebins = AgePartitions;

% the following parameters determine the standard deviation of the mu,
% lambda and alpha
alphamean = 0.0;
Aalpha=10;
Balpha=.5;
Ahyperalpha=10;
Bhyperalpha=.5;
Amu=5;
Bmu=.1;
Ahypermu=5;
Bhypermu=.1;
Alambda=10;
Blambda=.1;
Ahyperlambda=10;

```

```

Bhyperlambda=.1;

muvar = ones(iterationnum,1);
hypermubar=ones(iterationnum,1);
alphavar = ones(iterationnum,1); %Changed the alphavar in order to get a
% uniform induced prior for overall system reliability
hyperalphavar = ones(iterationnum,1);
lambdavar = ones(iterationnum,1);
hyperlambdavar = ones(iterationnum,1);

% ## seed is an integer which uniquely specifies the random number generator
seed = 1234;
%RandStream.setDefaultStream(RandStream('mt19937ar','seed',seed));

%ERROR CHECK
if iterationnum < 2
    error('iterationnum must be 2 or greater','Input Error');
    return;
end

if burnin > iterationnum
    error('burnin larger than iterationnum','Input Error');
    return;
end

munudge = 1.75;
lambdanudge = 0.75;
hyperlambdanudge = 1.0;
hypermunudge = 5; % Changed hypermunudge from .25 to 12
alphanudge = .15; % Changed alphanudge from .1 to .04
hyperalphanudge = .1; % Changed hyperalphanudge from .1 to .03
%% LOAD RESULTS CSV DATA

%HowLongIsThis = waitbar(0, 'Loading in test data');
ohno=0;

```

```

TestData = structCSV(filename);

%A= dataset('file',filename,'delimiter',' ','');

% if(isnumeric(TestData.Test)==0)
%     TestData.Test=cell2mat(TestData.Test);
% end
%
% if(isnumeric(TestData.Phase)==0)
%     TestData.Phase=cell2mat(TestData.Phase);
% end
% if(isnumeric(TestData.Component)==0)
%     TestData.Component=cell2mat(TestData.Component);
% end
% if(isnumeric(TestData.Mode)==0)
%     TestData.Mode=cell2mat(TestData.Mode);
% end

numtests = length(unique(TestData.Test)); % number of tests
numphase = length(unique(TestData.Phase)); % number of phases
numcomp = length(unique(TestData.Component)); % number of components
nummodes = length(unique(TestData.Mode)); % number of modes

% Here is where we implement the system only switch
if SystemOnly==1
    TestData = systemOnly(TestData);
    numphase=1; % need to change the numphase to 1
    numcomp=1; % need to change the numcomp to 1
end

N = length(TestData.Test); % number of rows in the test data
onevector = ones(N,1);
littleonevector = ones(numtests,1);

% the following matrices will summarize the success and fail information
% for each and every test result

```

```

% The following were made global variables because they were unnecessarily
% called in functions or passed between functions.

global successmatrix failmatrix unknownmatrix bigonevec isthereunknowns ...
    agematrix modematrix P T C MM whichUnknown
successmatrix = zeros(numcomp*numphase,numtests);
failmatrix = zeros(numcomp*numphase,numtests); %numcomp*numphase by numtests
unknownmatrix = zeros(numphase,numtests); %numphase by numtests
bigonevec = ones(1, numtests);
littleonevec = ones(1, numphase);
isthereunknowns = littleonevec*unknownmatrix;
T=numtests;

% These 4 matrices will summarize the covariate information we need for
% the model
modematrix = zeros(numtests,1); % numtests by 1
agematrix = zeros(numtests,1); % numtests by 1

% for each test determine which component failed and which phase
% In the following loops we will compile all the information on the
% covariates for the model (X's) and also the test result information (Y)
unknownindicator = strcmp(TestData.Comp_Result,'?',1);
successindicator = strcmp(TestData.Comp_Result,'1',1);
failindicator = strcmp(TestData.Comp_Result,'0',1);
modevector = TestData.Mode;
agevector = TestData.Age;

for t = 1: numtests
    ohno = ohno + 1;
    % waitbar(ohno/numtests);
    testindicator = (TestData.Test==t);
    modematrix(t) = max(unique(testindicator.*modevector));
    agematrix(t) = max(unique(testindicator.*agevector));
    for phase = 1:numphase
        phaseindicator = (TestData.Phase==phase);
        phasetestindicator = testindicator.*phaseindicator;
        isthereunknowns = unknownindicator.*phasetestindicator;
        unknowntally = isthereunknowns'*onevector;
    end
end

```

```

    if(unknownmatrix>0)
        unknownmatrix(phase,t)= 1;
    else
        unknownmatrix(phase,t) = 0;
    end
    for comp = 1:numcomp
        compindicator = (TestData.Component==comp);
        grandindicator = testindicator.*compindicator.*phaseindicator;
        successmatrix(numcomp*(phase-1)+comp,t)=successindicator'...
            *grandindicator;
        failmatrix(numcomp*(phase-1)+comp,t)=failindicator'...
            *grandindicator;
    end
end
end

%close(HowLongIsThis);

% all of the measures below aggregate the values over all modes
SS = successmatrix*ones(numtests,1); % successes by comp and phase
FF = failmatrix*ones(numtests,1); % known failures by comp and phase
UF = unknownmatrix*ones(numtests,1); % unknown failures by phase
failsbyphase = zeros(numphase,1); % contains failures by phase
testsbyphase = zeros(numphase,1); % will contain tests by phase
for phase = 1:numphase
    list= (numcomp*(phase-1)+1):(numcomp*phase);
    failsbyphase(phase) = sum( FF(list) );
    failsbyphase(phase) = failsbyphase(phase)+UF(phase);
    if(phase==1)
        testsbyphase(phase)=numtests;
    else
        testsbyphase(phase) = testsbyphase(phase-1) - failsbyphase(phase-1);
    end
end

end
successesbyphase=testsbyphase-failsbyphase;

```

```

%% LOAD PRIOR CSV DATA
%% ## We will now read in the prior data
PriorData = structCSV(priorfilename);

% note to self: we need to think about what would happen if M = 1!!

P = length(unique(PriorData.Phase)); % number of phases
C = length(unique(PriorData.Component)); % number of components
M = length(unique(PriorData.Mode)); % number of modes
PriorN = length(PriorData.Mode); % number of rows in priorfilename

PriorR = zeros(C*P,1);
PctEquiv = zeros((M-1)*C*P,1);
AgeEffect = zeros(C,1);

PI = zeros(C,P);
mu0 = zeros(C,P); %
PIstress = zeros(C,P,(M-1));
lambda0 = zeros(C,P,(M-1)); % this array will store logit(PIstress)

% This makes the unknow matrices for the case where we have unknown
% failures.

isthereunknowns=sum(unknownindicator);
unknownmatrix = zeros(numphase,numtests);
whichUnknown=find(sum(unknownmatrix)>0);

%HowLongIsThis = waitbar(0, 'Loading in prior data');
ohno=0;

for mode = 1: M
    modeind = (PriorData.Mode==mode);
    for phase = 1:P
        phaseind = (PriorData.Phase==phase);
        for comp = 1:C
            compind = (PriorData.Component==comp);
            grandind = modeind.*phaseind.*compind;

```



```

        if(mode == 1)
            PriorR(C*(phase-1)+comp)=grandind'...
                *(PriorData.PriorReliability);
        else
            PctEquiv(P*C*(mode-2)+C*(phase-1)+comp)=grandind'...
                *(PriorData.PctEquivilence);
        end

    end

end

end

end

% Load data into the Pi and mu0 array
% Do some error checking to prevent someone from putting PriorR = 1 or 0
% for some component.

for phase = 1:P
    for comp = 1:C
        j = C*(phase-1)+comp;
        if(PriorR(j)>0.9999999999)
            PriorR(j) = 0.9999999999;
        end
        if(PriorR(j)<0.001)
            PriorR(j) = 0.001;
        end
        PI(comp,phase)=PriorR(j);
        mu0(comp,phase)=log(PI(comp,phase))-log(1-PI(comp,phase));
    end
end

end

mu0=mean(mu0);

% ## For each mode, enter the number of pi values and the percent
% equivilency values. Also need to check if someone puts 0's in for
% PctEquiv, because both of these errors would be bad!!

for mode = 1:(M-1)
    ohno = ohno + 1;

```

```

% waitbar(ohno/M);
for phase = 1:P
    for comp = 1:C
        j = P*C*(mode-1)+C*(phase-1)+comp;
        if(PctEquiv(j)<10^(-10))
            PIstress(comp,phase,mode)= 0.9999999999;
            lambda0(comp,phase,mode) = log(PIstress(comp,phase,mode))...
                -log(1-PIstress(comp,phase,mode)); %
        else
            PIstress(comp,phase,mode) = 1.0 ...
                - PctEquiv(j)*(1.0 - PI(comp,phase));
            lambda0(comp,phase,mode) = log(PIstress(comp,phase,mode))...
                -log(1-PIstress(comp,phase,mode))-mu0(phase); .
        end
    end
end

end

end

end

MM = min(M,nummodes);
if(MM>1)
    needlambda=1;
else
    needlambda = 0;
end

% here are some error checkers
if (P > numphase)
    errordlg('number of phases in prior file is larger than in datafile'...,
        'Input Error');
    return;
end

if (P < numphase)
    errordlg('number of phases in prior file is less than in datafile'...
        , 'Input Error');
    return;
end

```

```

end

if (C < numcomp)
    errordlg('number of components in prior file is less than in datafile'...
        , 'Input Error');
    return;
end

if (C > numcomp)
    errordlg('number of components in prior file is greater than in datafile'...
        , 'Input Error');
    return;
end

if (nummodes > M)
    errordlg('number of test mode types in data file is greater than ...
        the number in the prior file', 'Input Error');
    return;
end

% ## We now need to initialize 3 arrays to store the data in.
% ## bigmu is the array needed to store the mu values
% ## biglambda is the the arrays needed to store the lambda values
% bigalpha stores the alpha values

bigmu = zeros(iterationnum,C,P);
hypermu = zeros(iterationnum,P); % hyperparameter for mu
muvec = zeros(iterationnum,C*P);

if(MM>1)
    biglambda = zeros(iterationnum,C,P,MM-1);
    hyperlambda = zeros(iterationnum,C,P,MM-1); % hyperparameter for lambda
    lambdavec = zeros(iterationnum,C*P*(MM-1));
else
    lambdavec = zeros(iterationnum,C*P);
end

bigalpha = zeros(iterationnum,C);
hyperalpha = zeros(iterationnum,1); % hyperparameters for alpha

```

```

for phase = 1:P
    for comp = 1:C
        if(testsbyphase(phase)>10)
            meann = (SS(C*(phase-1)+comp)/testsbyphase(phase)...
                + PI(comp,phase))/2.0;
            bigmu(1,comp,phase) = log(meann)-log(1-meann);
            % hypermu(1,comp,phase) = bigmu(1,comp,phase);
            muvec(1,C*(phase-1)+comp)=bigmu(1,comp,phase);
        else
            meann=PI(comp,phase);
            bigmu(1,comp,phase) = log(meann)-log(1-meann);
            % hypermu(1,comp,phase) = bigmu(1,comp,phase);
            muvec(1,C*(phase-1)+comp)=bigmu(1,comp,phase);
        end

        if(MM>1)
            for mode = 1:(MM-1)
                biglambda(1,comp,phase,mode) = lambda0(comp,phase,mode);
                hyperlambda(1,comp,phase,mode) = lambda0(comp,phase,mode);
                lambdavec(1,C*P*(mode-1)+C*(phase-1)+comp)...
                    =lambda0(comp,phase,mode);
            end
        else
            lambdavec(1,C*(phase-1)+comp)=0;
        end

    end
end
end

```

```

% These will count the number of acceptance and rejection decisions
% these will be utilized later on in the computation of the mix rate
muccount = 0;

```

```

lambdacount = 0;
alphacount = 0;

hypermucount = 0;
hyperlambdacount = 0;
hyperalphacount = 0;

%% MH Algorithm

ohno=0;

propalphavec=bigalpha(1,:);
oldalphavec=bigalpha(1,:);
promuvec = muvec(1,:);
oldmuvec = muvec(1,:);
proplambdavec = lambdavec(1,:);
oldlambdavec = lambdavec(1,:);

% Need to initialize a vector which will store the log-likelihood values
loglike = zeros(iterationnum,1);

% ## Initialize loop on Iteration in the Metropolis-Gibbs hybrid algorithm

for i = 2:iterationnum
    ohno = ohno + 1;

    %here is where we use the YesAge variable to "turn on/off" the age
    %effect

    muvar(i) = 1/gamrnd(Amu+(C*P)/2, 1/(1/Bmu+...
        (sum(sum((squeeze(bigmu((i-1),:,:))-...
            repmat(hypermu((i-1,:),[C,1])).^2))/2), [1 1]));
    hypermuvar(i)=1/gamrnd(Ahypermu+(C*P)/2, 1/(1/Bhypermu+...
        (sum((reshape(squeeze(hypermu((i-1),:)), size(mu0))-mu0).^2)/2)));

    if MM>1
        lambdavar(i) = 1/gamrnd(Alambda+(C*P*(MM-1))/2, 1/(1/Blambda ...

```

```

        + sum(sum(sum(((biglambda((i-1),:,:)-...
        hyperlambda((i-1),:,:)).^2),4),3),2)/2));
hyperlambdavar(i) = 1/gamrnd(Ahyperlambda+...
        (C*P*(MM-1))/2, 1/(1/Bhyperlambda...
        + sum(sum(sum((reshape(squeeze(hyperlambda((i-1),:,:))...size(lambda0))-lambda0).^2,3),2))/2
end

if YesAge==1

if DiffAge==1
%We start by updating the variance for alpha (alphavar).
alphavar(i)=1/gamrnd(Aalpha+C/2, 1/(1/Balpha+...
        (sum((bigalpha((i-1),:)-hyperalpha(i-1)).^2))/2), [1 1]);
hyperalphavar(i) = 1/gamrnd(Ahyperalpha+C/2, 1/(1/Bhyperalpha...
        +(hyperalpha(i-1)^2)/2), [1 1]);

%Here we initialize the next the vector for the ith iteration. This
%will allow us to get rid of these computationally costly if/elseif
%statements
bigalpha(i,:)=bigalpha((i-1),:);

%moved this outside the loop because it doesn't change
oldlambdavec = lambdavec((i-1),:);
oldmuvec = muvec((i-1),:);

for comp = 1:C
    alphaproposal = bigalpha((i-1),comp)+ alphanudge*(randn(1,1));
    % generate a proposal for alpha
    % form a proposal and old alpha vector

    propalphavec=bigalpha(i,:);
    oldalphavec=propalphavec;
    propalphavec(comp)=alphaproposal;

    logratio = loglikelihood(oldmuvec,oldlambdavec,propalphavec)...
        - loglikelihood(oldmuvec,oldlambdavec,oldalphavec);
    logratio = logratio + normlike([hyperalpha(i-1),...
        alphavar(i)],bigalpha((i-1),comp)) - ...
        normlike([hyperalpha(i-1),alphavar(i)],alphaproposal);

```

```

% generate a random uniform and take the log of it
U = log(rand(1,1));
if(U < logratio)
    bigalpha(i,comp) = alphaproposal; % accept the proposal
    alphacount = alphacount+1;
end

%We now update the hyperparameters for alpha using Gibbs
%sampling because they are conjugate priors for hyperalpha

end % we have now updated the parameters for alpha

hyperalpha(i)=sum(bigalpha(i,:))*...
hyperalphavar(i)/(C*hyperalphavar(i)...
+alphavar(i))+sqrt(hyperalphavar(i)*alphavar(i)/...
(C*hyperalphavar(i)+alphavar(i)))*randn(1,1);

else

alphavar(i)=1/gamrnd(Aalpha+C/2, 1/(1/Balpha+...
(sum((bigalpha((i-1),:)-hyperalpha(i-1)).^2))/2), [1 1]);
hyperalphavar(i) = 1/gamrnd(Ahyperalpha+C/2, 1/(1/Bhyperalpha+...
(hyperalpha(i-1)^2)/2), [1 1]);

%Here we initialize the next the vector for the ith iteration. This
%will allow us to get rid of these computationally costly if/elseif
%statements
bigalpha(i,:)=bigalpha((i-1),:);

%moved this outside the loop because it doesn't change
oldlambdavec = lambdavec((i-1),:);
oldmuvec = muvec((i-1),:);

alphaproposal = bigalpha((i-1),:)+ alphanudge*(randn(1,1));
% generate a proposal for alpha

```

```

% form a proposal and old alpha vector

oldalphavec=bigalpha(i,:);
propalphavec=alphaproposal;

logratio = loglikelihood(oldmuvec,oldlambdavec,propalphavec)...
    - loglikelihood(oldmuvec,oldlambdavec,oldalphavec);
logratio = logratio + normlike([hyperalpha(i-1),alphavar(i)],...
    bigalpha((i-1),comp)) - ...
    normlike([hyperalpha(i-1),alphavar(i)],alphaproposal);

% generate a random uniform and take the log of it
U = log(rand(1,1));
if(U < logratio)
    bigalpha(i,:) = alphaproposal; % accept the proposal
    alphacount = alphacount+1; %
end

%We now update the hyperparameters for alpha using Gibbs
%sampling because they are conjugate priors for hyperalpha

hyperalpha(i)=sum(bigalpha(i,:))*hyperalphavar(i)/(C*hyperalphavar(i)...
    +alphavar(i))+sqrt(hyperalphavar(i)*...
    alphavar(i)/(C*hyperalphavar(i)+alphavar(i)))*randn(1,1);

end

end %end if YesAge==1 statement

mode=1;
% we will now update the mu and lambda parameters
% the following loop structure does this

muvec(i,:)=muvec((i-1),:);
lambdavec(i,:)=lambdavec((i-1),:);

```



```

oldalphavec = bigalpha(i,:);

for phase = 1:P
    for comp = 1:C
        index = C*(phase-1)+comp;
        if(MM>1)
            lambdaindex =(mode-1)*C*P + C*(phase-1)+comp;
            endlambdaindex=C*P*(MM-1);
        else
            lambdaindex = C*(phase-1)+comp;
            endlambdaindex=C*P;
        end
        muproposal = bigmu((i-1),comp,phase)+ munudge*(randn(1,1));
        % form a proposal and old mu vector
        propmuvec=muvec(i,:);
        oldmuvec=muvec(i,:);
        propmuvec(index)=muproposal;

        proplambdavec=lambdavec(i,:);
        oldlambdavec=lambdavec(i,:);

        % evaluate the logratio of the posterior distributions = difference
        % in loglikelihood + difference in the logprior distributions
        logratio = loglikelihood(propmuvec,oldlambdavec,oldalphavec)...
            - loglikelihood(oldmuvec,oldlambdavec,oldalphavec);
        logratio = logratio + normlike([hyperm((i-1),phase),...
            sqrt(muvar(i))],bigmu((i-1),comp,phase))...
            - normlike([hyperm((i-1),phase),sqrt(muvar(i))],muproposal);
        % generate a random uniform and take the log of it
        U = log(rand(1,1));
        if(U < logratio)
            muvec(i,index) = muproposal; % accept the proposal
            mucount = mucount+1; % increment the mu acceptance counter
        end
        bigmu(i,comp,phase) = muvec(i,index);

        % done with hyperparameters for mu
        % ok the mu portion is done.
    end
end

```

```

if(MM>1)
    for mode = 1:(MM-1)
        index=(mode-1)*C*P + C*(phase-1)+comp;
        endindex=C*P*(MM-1);
        muindex= C*(phase-1)+comp;

        proplambdavec=lambdavec(i,:);
        oldlambdavec=lambdavec(i,:);

        lambdaproposal = lambdavec((i-1),index) + ...
            lambdanudge*(randn(1,1)); %
        proplambdavec(index)=lambdaproposal;

        logratio = loglikelihood(oldmuvec,proplambdavec,oldalphavec)...
            - loglikelihood(oldmuvec,oldlambdavec,oldalphavec);
        logratio = logratio + normlike([hyperlambda((i-1),...
            comp,phase,mode),lambdavar(i)],lambdavec((i-1),index))...
            - normlike([hyperlambda((i-1),comp,phase,mode),...
            lambdavar(i)],lambdaproposal);
        % generate a random uniform and take the log of it
        U = log(rand(1,1));
        if(U < logratio)
            lambdavec(i,index) = lambdaproposal;
            lambdacount = lambdacount+1;
        end

        biglambda(i,comp,phase,mode)=lambdavec(i,index);

        % now update the hyperparameters for lambda
        % done with hyperparameters for mu

    end % this end statement belongs with the mode loop

end % this ends the if loop around the lambda update portion

```

```

    end % end the loop on component

end % end the loop on phase

hypermui(i, :)=((squeeze(sum(bigmui(i, :, :))) * hypermuvar(i) + ...
    mu0' * muvar(i)) / (muvar(i) ...
    + C * hypermuvar(i)) + sqrt((muvar(i) * hypermuvar(i)) / ...
    (C * hypermuvar(i) + muvar(i))) * randn(size(mu0'))');

if MM > 1
    hyperlambdai(i, :, :, :)= (reshape(biglambdai(i, :, :, :), size(lambda0)) * ...
        hyperlambdavar(i) + lambda0(:, :, :)) * lambdavar(i) / ...
        (hyperlambdavar(i) + lambdavar(i)) ...
        + sqrt((hyperlambdavar(i) * lambdavar(i)) / (hyperlambdavar(i) + ...
        lambdavar(i))) * randn(size(lambda0)));
end

% We should store the value of loglikelihood(i)
% the two cases where MM = 1 and MM > 1 turn out to both be satisfied
% with this 1 command!!!

loglike(i) = loglikelihood(muvec(i, :), lambdavec(i, :), bigalpha(i, :));

% we will collect the loglikelihood values in this one vector so that
% they can be used in the computation of DIC and the effective # of
% parameters

if rem((i - burnin), 100000) == 0
    workspacefilename = sprintf('%s/PosteriorData/%s-%i-Model%i-Iter%i', ...
        cd, testdata, iterationnum, modelnum, i);
    save(workspacefilename);
end
if rem(i, 10000) == 0
    workspacefilename = sprintf('%s/PosteriorData/%s-%i-Model%i-iternum%i', ...
        cd, testdata, iterationnum, modelnum, i);
    save(workspacefilename, 'i');
end

```

```

end % end the loop on iterationnumber
%close(HowLongIsThis);
% Yeah we are done with the Metropolis-Gibbs loop!!!! Hooorrrraaaahhh!!!

toc(timeElapsed);
MCMCtime=toc(timeElapsed);
workspacefilename=sprintf('%s/PosteriorData/%s-%i-Model%i',cd,testdata,...
    iterationnum,modelnum);
save(workspacefilename);

end

```

A.2 LIKELIHOOD FUNCTION

```

function [loglike] = loglikelihood(muvec,lambda,alpha)

% This function will compute the loglikelihood for a given parameter vector

% The parameter space Capital Theta consists of the parameters mu
% (comp,phase) which represents the mean reliability for the component,
% phase
% when the missile is undergoing flight (or usage testing). By
% convention we will take mode = 1 to represent flight testing.

% The parameter lambda(comp,phase,mode) represents the effect on the
% reliability due to testing the missile at stress levels other than
% flight stress levels (all modes > 1).

% The parameter alpha will represent the age effect on each component. We
% assume that the age effect on reliabiltiy for each component will be
% constant
% for different phases and different test modes.

% Let P = number of missile phases
% Let C = number of components
% Let M = number of test modes
% Let T = the number of tests conducted

```

```

% muvec will be a vector having dimension ( C*P by 1)
% lambda will be a vector having dimension ( C*P*(M-1) by 1)
% alpha will be a vector having dimension (C by 1)

% the loglikelihood function is also a function of the test data which
% includes the covariate information (X variables) and the test result
% information (Y's). The Y variables in this model are success and fail
% information by component and phase. This Y variable information is
% stored in the matrices successmatrix, failmatrix, and unknownmatrix.

% Here is the Y variable information:
% unknownmatrix is a (P by T) matrix of 1's or 0's which tell us whether
% unknown failures occurred by component
% successmatrix is a (C*P by T) binary matrix for successes by comp and
% phase
% failmatrix is a (C*P by T) binary matrix for failures by comp and phase

% Here is the X variable information:
% modematrix is a (T by 1) recording the test mode by test
% agematrix is a (T by 1) recording the age by test

% the linear predictor is everything to the right hand side of the logit
% equation. Specifically the linear predictor for this model is given by
% Linear Predictor = mu(i,j) + lambda(i,j,k)*I(k>1) + alpha(i)*Age +
% beta(i)* I(maintenance >0) + gamma(j) * (mission time)

%The following variables are called globally to prevent them from being
%called/passed when they don't need to be. Also, because we changed these
%to global variables, the function loglikelihood now only needs to pass 3
%variables.

%We also made P T and C global

global successmatrix failmatrix unknownmatrix bigonevec isthereunknowns ...
    agematrix modematrix P T C MM whichUnknown

% First we convert the lambda matrix into a form where it is easy to add to
% the linear predictor matrix.

```

```

lambdaByTest=0;

if MM>1
    lambdamatrix=[zeros(1,(P*C)) lambda]; % add zeros to the front of the
%   lambdas. This lets us add 0 if the mode of the test is 1.
    lambdamatrix=reshape(lambdamatrix, P*C, MM); % we reshape the matrix
%   in order to get it in the same dimensions as the elements in the linear
%   predictor.
    lambdamatrix=lambdamatrix'; % Because the reshape function enters
%   things in by column, and we want them inserted in rows, we transpose
%   the matrix.
    lambdaByTest=lambdamatrix(modematrix, :); % now we take the row of the
%   lambda matrix that corresponds to the mode of each test.
end

% This calculates the linearpredictor for each comp, phase, mode

% The first line just adds the mu's, alpha's, and lambda's for each
% component and phase for each test. We then use the transform to get logR
% and log1minusR
linearpred= repmat(muvec, [T,1]) + repmat((agematrix*alpha), [1,P])...
    +lambdaByTest;
logR=linearpred-log1p(exp(linearpred));
log1minusR=(-1)*log1p(exp(linearpred));

% Using the successmatrix and the failmatrix, we calculate the
% loglikelihood. We take the diagonal of the matrix multiplication because
% we only need the cross product of the logR for test t, and successmatrix
% for test t. So we only want the (t,t)th entry of logR*successmatrix.
loglikelihood= logR*successmatrix + log1minusR*failmatrix;
loglikelihood=diag(loglikelihood);

loglike = bigonevec*loglikelihood;
return;

```