I^2SDS
The Institute for Integrating Statistics in Decision Sciences

Technical Report TR-2007-8
May 22, 2007

Advances in Bayesian Software Reliability Modelling

Fabrizio Ruggeri
CNR IMATI
Milano, Italy

Refik Soyer
Department of Decision Sciences
The George Washington University
Washington, DC

Advances in Bayesian Software Reliability Modelling

Fabrizio RUGGERI [a,1], and Refik SOYER [b]
[a] *CNR IMATI, Milano, Italy*
[b] *Department of Decision Sciences, The George Washingon University, USA*

Abstract. This paper reviews recent developments in Bayesian software reliability modeling. In so doing, emphasis is given to two models which can incorporate the case of reliability deterioration due to potential introduction of new bugs to the software during the development phase. Since the introduction of bugs is an unobservable process, latent variables are introduced to incorporate this characteristic into the models. The two models are based, respectively, on a hidden Markov model and a self-exciting point process with latent variables.

Keywords. Hidden Markov models, reliability growth, Markov chain Monte Carlo, self-exciting point process

Introduction

Many papers have been published on software reliability during the last several decades; see Jelinski and Moranda (1972) and Musa and Okumoto (1984) as examples of early work. Bayesian methods have been widely used in this field as discussed in Singpurwalla and Wilson (1999). In this paper we plan to review some of the Bayesian models introduced recently focussing especially on our ongoing research. We present two models that are motivated by potential introduction of new bugs to the software when fixing the current ones. The first model, based on a hidden Markov chain, assumes that times between failures are exponentially distributed with parameters depending on an unknown latent state variable which, in turn, evolves as a Markov chain. The second model considers a self-exciting point process whose intensity might increase each time a bug is attempted to be fixed. Unobserved outcomes of latent Bernoulli random variables are introduced to model the possible introduction of new bugs and the consequent increase in the intensity function of the process. Both models take in account the possibility of not knowing if a new bug has been added at each stage and they can be applied not only to model the failure process but also to infer if new bugs were introduced at different testing stages.

In Section 1 we will review some of the earlier work on potential introduction of new bugs to the software during the debugging phase. In Section 2 we will describe the hidden Markov model (HMM) and will apply it to the Jelinski and Moranda's Naval Tactical data and Musa's System 1 data in Section 3. The self-exciting point process

[1]Corresponding Author: Fabrizio Ruggeri, CNR IMATI, Via Bassini 15, I-20133 Milano, Italy; E-mail: fabrizio@mi.imati.cnr.it.

(SEP) with latent variables will be described in Section 4. Discussion on current research will be presented in Section 5.

1. Earlier work on imperfect debugging

Although the models introduced in this paper are novel ones, possibility of imperfect debugging and introduction of new bugs during software testing have been considered in earlier papers. Here we review some of them.

Gaudoin, Lavergne and Soler (1994) considered failures at times $T_1 < \ldots < T_n$ and modelled the interfailure times with independent exponential distributions. In particular, they took

$$T_i - T_{i-1} \sim \mathcal{E}(\lambda_i), i = 1, \ldots, n.$$

with

$$\lambda_{i+1} = \lambda_i e^{-\theta_i}, \tag{1}$$

where λ_i and θ_i, $i = 1, \ldots, n$, are nonnegative. From (1), it is clear that the parameter θ_i plays a relevant role in describing the effect of the intervention during software testing. If $\theta_i = 0$, then there is no debugging effect on software reliability, which increases (decreases) if $\theta_i > 0$ ($\theta_i < 0$). The latter case is due to introduction of new bugs to the software.

A slightly modified version of this model was proposed by Gaudoin (1999), who considered

$$\lambda_{i+1} = (1 - \alpha_i - \beta_i)\lambda_i + \mu\beta_i,$$

for modelling the more realistic case where intervention at each stage may introduce new bugs while fixing the existing ones at the same time. The effect of the positive intervention is modelled by α, whereas β is used for the negative one.

A different model to address the same issue was originally proposed by Kremer (1983) who considered a birth-death process $X(t)$ denoting the number of bugs in the software at time t. Starting with $X(0) = a$, then $p_n(t) = \mathcal{P}r\{X(t) = n\}$ is obtained as the solution of the differential equation

$$p_n^{'}(t) = (n-1)\nu(t)\,p_{n-1}(t) - n\nu(t) + \mu(t)\,p_n(t) + (n+1)\mu(t)\,p_{n+1}(t), n \geq 0,$$

with $p_{-1} \equiv 0$ and $p_n(0) = 1(n = a)$, where $1(\cdot)$ is the indicator function. Here $\nu(t)$ (birth rate) and $\mu(t)$ (death rate) denote, respectively, the rate of introduction of new bugs and the rate of fixing of old ones.

More recently, Durand and Gaudoin (2005) considered a hidden Markov model similar to the one we introduce in Section 2, but they considered nonBayesian approach and used an EM algorithm to obtain maximum likelihood estimates. They applied the Bayesian information criterion (BIC) to choose among models with different number of states of the hidden process.

2. A hidden Markov model for software failures

We assume that, during the testing stages, the failure rate of the software is governed by a latent process Y. Let Y_t denote the state of the latent process at time t and, given the state at time t is i, assume that, X_t, the failure time for period t follows an exponential model given by

$$X_t|Y_t = i \sim \mathcal{E}(\lambda(i)).$$

The states of the latent process reflect the effectiveness of the interventions, i.e. the design changes, to the software prior to the t-th stage of testing. The failure rate of the software depends on this latent random variable.

We assume that the latent process $Y = \{Y_t \quad t \geq 1\}$ is a Markov chain with a transition matrix P on a finite state space $E = \{1, \ldots, k\}$. Given the latent process, we assume that X_t's are conditionally independent, that is,

$$\pi(X_1, X_2, \ldots, X_n|Y) = \prod_{t=1}^{n} \pi(X_t|Y).$$

In the Bayesian setup we assume that the transition matrix P and the failure rate $\lambda(i)$, for $i = 1, \ldots, k$, are all unknown quantities. For the components of the transition matrix, it is assumed that $P_i = (P_{i1}, \ldots, P_{ik})$, $i = 1, \ldots, k$, i.e. the i-th row of P, follows a Dirichlet distribution $\mathcal{D}ir(\alpha_{i1}, \ldots, \alpha_{ik})$, as

$$\pi(P_i) \propto \prod_{j=1}^{k} P_{ij}^{\alpha_{ij}-1} \tag{2}$$

with parameters α_{ij}, $i, j = 1, \ldots, k$, and such that the P_i's are independent of each other. For a given state $i = 1, \ldots, k$, we assume a Gamma prior

$$\lambda(i) \sim \mathcal{G}(a(i), b(i)),$$

with independent $\lambda(i)$'s.

If software failures are observed for n testing stages, then, given the observed data $x^{(n)} = (x_1, x_2, \ldots, x_n)$, we are interested in the joint posterior distribution of all unknown quantities $\Theta = (\lambda^{(n)}, P, Y^{(n)})$, where $\lambda^{(n)} = (\lambda(1), \ldots, \lambda(n))$, and $Y^{(n)} = (Y_1, \ldots Y_n)$. It is not computationally feasible to evaluate the joint posterior distribution of Θ in closed form. However, we can use a Gibbs sampler to draw samples from the joint posterior distribution.

The likelihood function is

$$\mathcal{L}(\Theta; x^{(n)}) = \prod_{t=1}^{n} \lambda(Y_t) e^{-\lambda(Y_t)\, x_t}$$

and the posterior distribution is given by

$$\pi(\Theta|x^{(n)}) \propto \Big[\prod_{t=1}^{n} P_{Y_{t-1},Y_t}\, \lambda(Y_t) e^{-\lambda(Y_t)\, x_t}\Big] \Big[\prod_{i=1}^{k} \pi(P_i)\, \lambda(i)^{a(i)-1}\, e^{-b(i)\lambda(i)}\Big],$$

where $\pi(P_i)$ is given by (2). The implementation of the Gibbs sampler requires draws from the full conditional distributions of the unknown quantities, that is, the components of Θ. We first note that, given $Y^{(n)}$, the full conditional distribution of the elements of P can be obtained as

$$P_i|Y^{(n)} \sim Dir\{\alpha_{ij} + \sum_{t=1}^{n} 1(Y_t = i, Y_{t+1} = j);\ j \in E\} \tag{3}$$

where $1(\cdot)$ is the indicator function and, given $Y^{(n)}$, P_i's are obtained as independent Dirichlet vectors. Given $Y^{(n)}$, they are also independent of other components of Θ.

The full conditional posterior distribution of $\lambda(i)$'s can be obtained as

$$\lambda(i)|Y^{(n)},\ x^{(n)} \sim \mathcal{G}(a^*(i), b^*(i)) \tag{4}$$

where

$$a^*(i) = a(i) + \sum_{t=1}^{n} 1(Y_t = i)$$

and

$$b^*(i) = b(i) + \sum_{t=1}^{n} 1(Y_t = i)\, x_t.$$

Finally, we can show that the full conditional posterior distributions of Y_t's are given by

$$\pi(Y_t|Y^{(-t)},\ \lambda(Y_t),\ x^{(n)},\ P) \propto P_{Y_{t-1},Y_t}\ \lambda(Y_t) e^{-\lambda(Y_t)\, x_t} P_{Y_t,Y_{t+1}} \tag{5}$$

where $Y^{(-t)} = \{Y_s; s \neq t\}$. Note that the above is a discrete distribution with constant of proportionality given by

$$\sum_{j \in E} P_{Y_{t-1},\, j}\ \lambda(j)\, e^{-\lambda(j)\, x_t} P_{j,\, Y_{t+1}}.$$

Thus, we can draw a posterior sample from $\pi(\Theta|x^{(n)})$ by iteratively drawing from the given full conditional posterior distributions. If we start with an initial value of the states, say, $Y_0^{(n)}$, then we can update the probability transition matrix via (3). Then, given the data and $Y_0^{(n)}$, we can draw the failure rates independently using (4). Given these values, we can use (5) to draw a new sample for the states. We can repeat these iterations many times to obtain a joint posterior sample.

Posterior predictive distribution of X_{n+1}, after observing $x^{(n)}$, is given by

$$\pi(X_{n+1}|x^{(n)}) = \sum_{j \in E} \int \pi(X_{n+1}|\lambda(j))\, P_{Y_n,j}\ \pi(\Theta|\, x^{(n)})\ d\Theta,$$

which can be approximated as a Monte Carlo integral via

$$\pi(X_{n+1}|x^{(n)}) \approx \frac{1}{G}\sum_{g=1}^{G}\pi(X_{n+1}|\lambda^g(Y_{n+1}^g)),$$

where Y_{n+1}^g is sampled given the posterior sample Y_n^g, using Dirichlet probabilities P_{Yg} given by (3).

3. Analysis of software reliability data

We next illustrate the use of the HMM by applying it to two well known datasets, the Jelinski and Moranda's Naval Tactical data and Musa's System 1 data.

3.1. Jelinski-Moranda data

The data, presented in Jelinski and Moranda (1972), consists of 34 failure times (in days) of a large military system, and is referred to as the Naval Tactical Data System (NTDS). In the analsis of the NTDS data, we consider two possible states for Y_t, i.e. $E = \{1, 2\}$ and assume uniform distributions for the rows P_i, $i = 1, 2$, of the transition matrix. We describe uncertainty about the λ's, by considering diffuse priors $\lambda(i) \sim \mathcal{G}(0.01, 0.01)$, $i = 1, 2$. Gibbs sampler was run for 5000 iterations and no convergence problems were observed. In what follows we present the posterior results for major quantities of interest as illustrated by plots and tables.

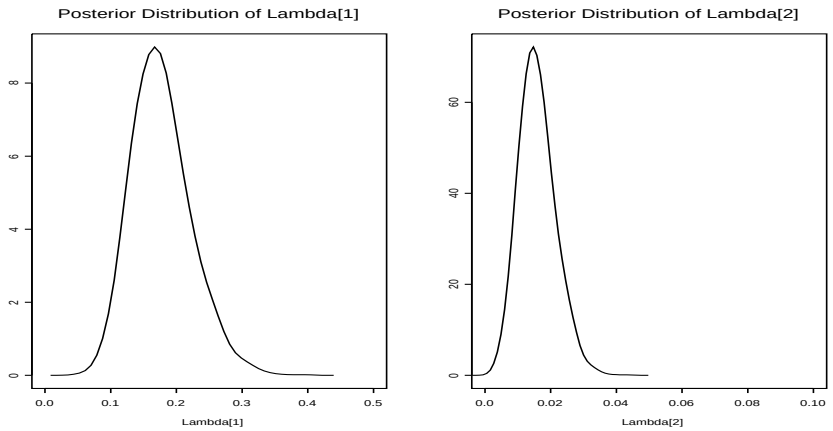

Figure 1. Posterior distributions of $\lambda(1)$ and $\lambda(2)$.

In Figure 1 we present the posterior distributions of λ_1 and λ_2. As can be seen from Figure 1, the posterior distribution of λ_1 is concentrated at higher values than that of λ_2 implying that environment 1 is the less desirable of the two environments. In other words, it represents the environment with higher failure rates and smaller expected time to failures.

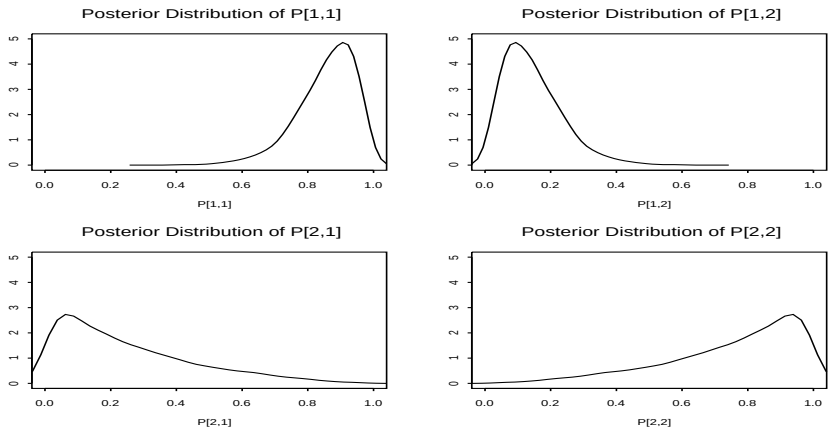

Figure 2. Posterior distributions of transition probabilities.

Posterior distributions of transition probabilities are presented in Figure 2. We can see from Figure 2 that the process Y_t tends to stay in environment 1 (compared to environment 2) from one testing stage to the next one. This is implied by the posterior distribution of P_{11} which is concentrated around values that are higher than 0.6. Posterior predictive distribution of the next time to failure, that is, the distribution of X_{35} is shown in Figure 3. As we can see from the predictive density, the next time to failure is expected within few days.

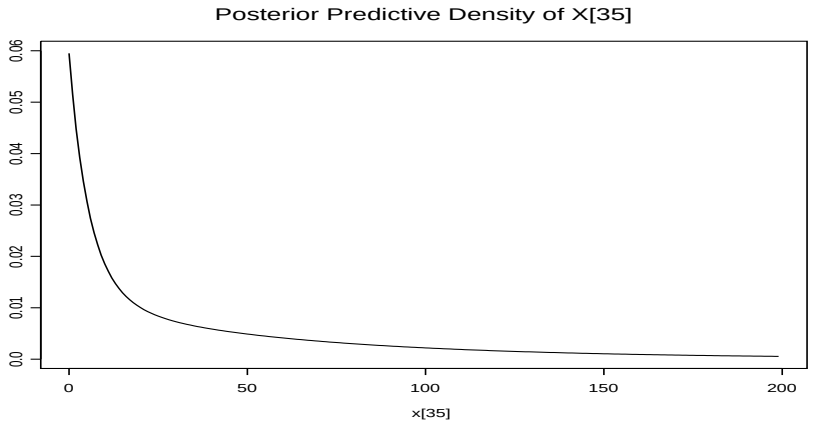

Figure 3. Predictive distribution of 35-th observation.

Table 1 presents the posterior distributions of the environment 1 for time periods, $t = 1, \ldots, 34$ as well as the observed time to failures for the periods. As we can see

Table 1. Posterior probabilities of state 1 over time.

t	X_t	$P(Y_t = 1\|D)$	t	X_t	$P(Y_t = 1\|D)$	t	X_t	$P(Y_t = 1\|D)$
1	9	0.8486	2	12	0.8846	3	11	0.9272
4	4	0.9740	5	7	0.9792	6	2	0.9874
7	5	0.9810	8	8	0.9706	9	5	0.9790
10	7	0.9790	11	1	0.9868	12	6	0.9812
13	1	0.9872	14	9	0.9696	15	4	0.9850
16	1	0.9900	17	3	0.9886	18	3	0.9858
19	6	0.9714	20	1	0.9584	21	11	0.7100
22	33	0.2036	23	7	0.3318	24	91	0.0018
25	2	0.6012	26	1	0.6104	27	87	0.0020
28	47	0.0202	29	12	0.2788	30	9	0.2994
31	135	0.0006	32	258	0.0002	33	16	0.1464
34	35	0.0794						

from the Table the posterior probability of the ”bad” environment (i.e. environment 1) decreases as we observe longer failure times.

3.2. Musa's System 1 data

We next consider the System 1 data of Musa (1979) which consists of 136 software failure times. As in the case of the Jelinski-Moranda data, we consider only two states for Y_t, and assume uniform distributions for the row vectors P_i of the transition matrix, and the same diffuse gamma distributions for the λ's. As before 5000 iterations of the Gibbs sampler was run and this led to convergence for all the quantities. The posterior analysis for the major quantities of interest will be presented in the sequel using few plots.

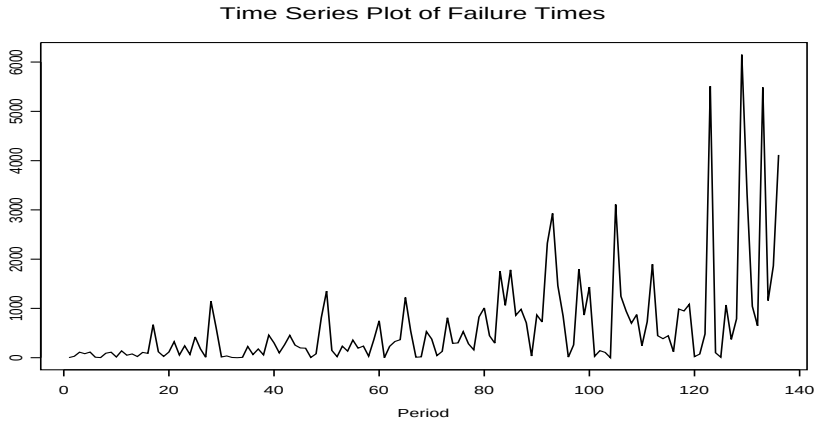

Figure 4. Failure times.

From Figure 4, we can see that the times between failures tend to increase over time implying an overall reliability growth. The posterior distributions of the λ_1 and λ_2 are

presented in Figure 5. We can see from Figure 5 that the posterior distribution of λ_1 is concentrated around lower values than that of λ_2. Thus environment 1 is the more desirable of the two environments, that is, it represents the environment with smaller failure rates and larger expected time to failures. In Figure 6 we present the posterior distributions of transition probabilities. We can see from the figure that the process Y_t tends to stay in the same state from one testing stage to the next one. Posterior predictive distribution of the next time to failure, that is, the distribution of X_{137} is shown in Figure 7. As can be seen from the figure the time to the next failure in this case has more variability than the one in the Jelinski-Moranda data shown in Figure 3.

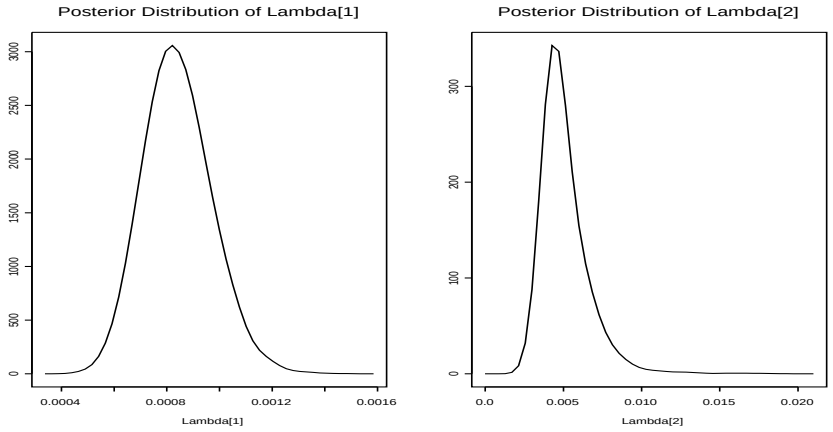

Figure 5. Posterior distributions of $\lambda(1)$ and $\lambda(2)$.

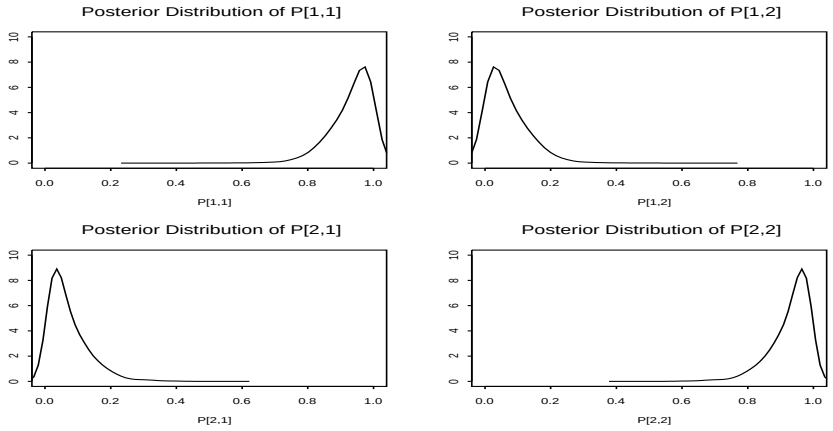

Figure 6. Posterior distributions of transition probabilities.

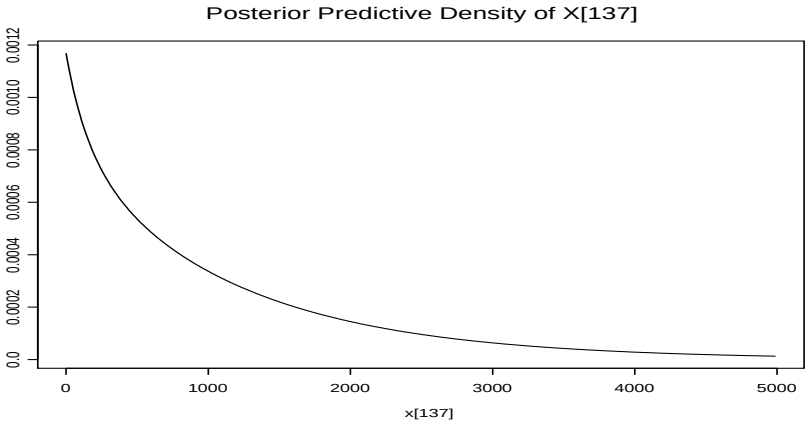

Figure 7. Predictive distribution of 137-th observation.

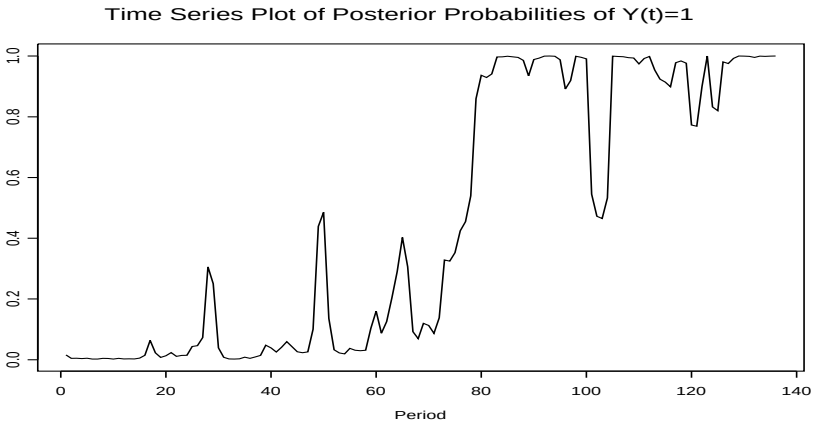

Figure 8. Posterior probability of $Y_t = 1$.

In Figure 8 we present the posterior probabilities $P(Y_t = 1|D)$ for the "good" environment, that is, for environment 1, for time periods $t = 1, \ldots, 136$. As we can see from the figure, the posterior probability is rather low for most of the first 80 testing stages implying that modifications which are made to the software during these stages have not improved the reliability from one period to the next. On the other hand, the posterior probabilities for environment 1 wander around values higher than 0.85 for most of the stages implying the improvement in the reliability achieved during the later stages. We note that as in the case of the Jelinski-Moranda data, the higher posterior probabilities in Figure 8 are associated with longer failure times shown in Figure 4.

4. Self-exciting point process with latent variables

Self-exciting point processes have an important role in software reliability since they can be used to unify existing models into a unique class as shown by Chen and Singpurwalla (1997). In this section, we consider a self-exciting process with latent variables that enables us to infer if a new bug has been introduced at each testing stage and the process intensity has increased.

We consider a nonhomogeneous Poisson process (NHPP) with intensity function $\mu(t)$ to describe the behaviour of the software when no bugs are added at each testing phase. We assume that the intensity is modified by positive valued functions $g(t - t_i)$ at each testing phase i as a consequence of the introduction of a new bug. We introduce Bernoulli random variables Z_j's to describe the introduction of a new bug during the i-th testing phase. As a consequence we consider a self-exciting point process (SEP) with latent variables with intensity

$$\lambda(t) = \mu(t) + \sum_{j=1}^{N(t^-)} Z_j g_j(t - t_j),$$

where $\mu(t)$ is the intensity of process without introduction of new bugs and $N(t^-)$ is the number of failures right before t, $t_1 < t_2 < \ldots < t_n$ are the failures in $(0, T$. The latent variable $Z_j = 1$ if a bug is introduced after the j–th failure and $Z_j = 0$ otherwise, and the function $g_j(u) \geq 0$ for $u > 0$ and $= 0$ otherwise.

Under these assumptions the likelihood function is given by $L(\theta; t^{(n)}, Z^{(n)}) = f(t^{(n)}|Z^{(n)}, \theta) f(Z^{(n)}|\theta)$, where $t^{(n)} = (t_1, t_2, \ldots, t_n)$ and $Z^{(n)} = (Z_1, Z_2, \ldots, Z_n)$ with

$$\begin{aligned} f(t^{(n)}|Z^{(n)}, \theta) &= \prod_{i=1}^{n} \lambda(t_i) e^{-\int_0^T \lambda(t)dt} \\ &= \prod_{i=1}^{n} \left[\mu(t_i) + \sum_{j=1}^{i-1} Z_j g(t_i - t_j) \right] e^{-\int_0^T \mu(t)dt - \sum_{j=1}^{N(T^-)} Z_j \int_0^{T-t_j} g_j(t)dt}, \end{aligned}$$

and dependence on θ is suppressed. In our analysis we consider the Power Law process (PLP) with intensity function $\mu(t) = M\beta t^{\beta-1}$, with $M > 0$ and $\beta > 0$. We assume also that $\mu \equiv g_j$, for all j, i.e. the contribution of each new bug is represented by the same PLP as the baseline process.

In this case we obtain

$$\begin{aligned} f(t^{(n)}|Z^{(n)}, \theta) &= M^n \beta^n \prod_{i=1}^{n} \left[t_i^{\beta-1} + \sum_{j=1}^{i-1} Z_j (t_i - t_j) \right] e^{-M\left[T^\beta + \sum_{j=1}^{N(T^-)} Z_j (T - t_j)^\beta\right]} \\ &= M^n \beta^n \prod_{i=1}^{n} A_i(\beta, Z^{(i-1)}) e^{-MB(\beta, Z^{(n)})}, \end{aligned}$$

where $Z^{(i)} = (Z_1, \ldots, Z_i)$, $A_i(\beta, Z^{(i-1)}) = t_i^{\beta-1} + \sum_{j=1}^{i-1} Z_j(t_i - t_j)$ and $B(\beta, Z^{(n)}) = T^\beta + \sum_{j=1}^{N(T^-)} Z_j(T - t_j)^\beta$. Considering $Z_j \sim \mathcal{B}ern(p_j)$, for all j, then it follows that

$$f(t^{(n)}, Z^{(n)}|\theta) = f(t^{(n)}|Z^{(n)}, \theta) f(Z^{(n)}|\theta) = f(t^{(n)}|Z^{(n)}, \theta) \prod_{j=1}^{n} p_j^{Z_j} (1 - p_j)^{1-Z_j}.$$

Given the likelihood function the two plausible strategies are either summing over all $Z^{(n)}$ so that $f(t^{(n)}|\theta)$ can be obtained or treating Z_j's as parameters and using MCMC methods. We follow the latter approach. We assume the prior distributions as $M \sim \mathcal{G}(\alpha, \delta)$, $\beta \sim \mathcal{G}(\rho, \lambda)$ and $p_j \sim \mathcal{B}eta(\mu_j, \sigma_j)$, for all j. Other possibilities about p_j could be an autoregressive model based on logit(p_j), a more general Markov chain or to use a common distribution $\mathcal{B}eta(\mu, \sigma)$, for all j.

We define $p^{(n)} = (p_1, \ldots, p_n)$, $p_{-j} = (p_1, \ldots, p_{j-1}, p_{j+1}, \ldots, p_n)$ and $Z_{-j} = (Z_1, \ldots, Z_{j-1}, Z_{j+1}, \ldots, Z_n)$. Also, we suppress the dependence on $t^{(n)}$. The full posterior conditionals are given by

- $M|\beta, Z^{(n)}, p^{(n)} \sim \mathcal{G}(\alpha + n, \delta + B(\beta, Z^{(n)})$
- $\beta|M, Z^{(n)}, p^{(n)} \propto \beta^{\rho+n} \prod_{i=1}^{n} A_i(\beta, Z^{(i-1)}) e^{-MB(\beta, Z^{(n)}) - \lambda\beta}$
- $p_j|M, \beta, Z^{(n)}, p_{-j} \sim \mathcal{B}eta(\mu_j + Z_j, \sigma_j + (1 - Z_j)), \forall j$

It follows from the above that

$$\mathbf{P}(Z_j = r|M, \beta, p^{(n)}, Z_{-j}) = \frac{C_r}{C_0 + C_1}, r = 0, 1,$$

with

$$C_0 = \prod_{i=j+1}^{n} \left[t_i^{\beta-1} + \sum_{h=1, i-1; h\neq j} Z_h (t_i - t_h)^\beta \right]$$

and

$$C_1 = \prod_{i=j+1}^{n} \left[t_i^{\beta-1} + \sum_{h=1, i-1; h\neq j} Z_h (t_i - t_h)^\beta + (t_i - t_j)^\beta \right] e^{-M(T-t_j)^\beta}.$$

Thus, we can draw a posterior sample from the joint distribution by iteratively drawing from the given full conditional posterior distributions.

5. Discussion

Possible extensions of the above models are currently under consideration. For example, in the HMM the dimension of the state space of the Markov chain will be typically unknown and this can be incorporated into the model as another random quantity. Other possible extensions include a dynamic evolution of the $\lambda(i)$'s, a nonhomogeneous Markov chain for the states of the latent process Y_t. Specification of a prior distribution

for the initial environment Y_0, which has been assumed as given here and estimation of the stationary distribution of the Markov chain are other issues under consideration.

Regarding the SEP model we are aware that the PLP is not most appropriate choice in this context and that other NHPP's with finite intensities should be explored. Therefore, we plan to consider different baseline processes, possibly in the family of the NHPP's whose intensity function can be written as $\mu(t) = Mg(t;\beta)$. Posterior analysis under these NHPP's are very similar to the ones obtained with the PLP as discussed in Ruggeri and Sivaganesan (2005).

An alternate class of models to what we consider here is NHPPs with change points as discussed in Ruggeri and Sivaganesan (2005). Other considerations include analysis with different actual software failure data sets and development of optimal testing policies.

References

[1] Y. Chen and N. D. Singpurwalla, Unification of software reliability models via self-exciting point processes,*Advances in Applied Probability* textbf29 (1997), 337–352.

[2] J.-B. Durand and O. Gaudoin, Software reliability modelling and prediction with hidden Markov chains, *Statistical Modelling* **5** (2005), 75–93.

[3] O. Gaudoin, Software reliability models with two debugging rates, *International Journal of Reliability, Quality and Safety* **6** (1999), 31–42.

[4] O. Gaudoin, C. Lavergne and J. L. Soler, A generalized geometric de-eutrophication software-reliability model, *IEEE Transactions on Reliability* **R-44** (1994), 536–541.

[5] Z. Jelinski and P. Moranda, Software reliability research, *Statistical Computer Performance Evaluation*,W. Freiberger (Ed.), (1972). New York: Academy Press.

[6] W. Kremer, Birth-death and bug counting, *IEEE Transactions on Reliability* **R-32** (1983), 37–46.

[7] J. D. Musa, Software reliability data, *Technical Report* (1979), Rome Air Development Center.

[8] J. D. Musa and K. Okumoto, A logarithmic Poisson execution time model for software reliability measurement, *Proceedings of the seventh International Conference on Software Engineering* 1984, 230–237.

[9] F. Ruggeri and S. Sivaganesan, On modeling change points in nonhomogeneous Poisson processes, *Statistical Inference for Stochastic Processes* **8** (2005), 311–329.

[10] N. D. Singpurwalla and S. Wilson, *Statistical Methods in Software Engineering*, Springer Verlag, New York, 1999.