解　説

ソフトウェアの信頼性におけるベイズ統計モデル†

三　道　弘　明††

1. はじめに

本特集でこれまでにみてきたように，ソフトウェアの信頼性を定量的に捉えることを目的として開発されたモデルにはさまざまなものが存在する．特に開発段階に注目した場合，テストとデバッグが繰り返されるにともなって，ソフトウェアの信頼性は通常向上すると考えられる．この信頼性が向上する様子をモデル化したものが信頼度成長モデルと呼ばれるものであることは，本特集でこれまでにも述べられてきたとおりである．本稿ではこの信頼度成長モデルのうちで，ベイズ統計学の立場から考案されたモデルを紹介する．

ベイズ統計学では，ソフトウェア，ハードウェアに関係なく，種々のモデルにおいて本来確定変数であるパラメータを確率変数として取り扱うことにより，その不確実性を定量的に捉える．ハードウェアの信頼性においては，このようなベイズ統計学の考え方を応用した研究が古くからなされており，数多くの研究成果が報告されている[1]．

ソフトウェアの信頼性においても，ベイズ統計学の考え方を応用したものがいくつか報告されており[2)-18)]，中でも Littlewood ら[2),3),7)]のモデルは欧米で積極的に利用されている．以下では，彼らの提案したモデルに焦点を絞り，ソフトウェアの信頼性におけるベイズ論的モデルについて概観する．

2. ソフトウェアの信頼性に関するベイズ論的モデル

本章では，初めにベイズ統計学の基本的考え方を概説する．次いで，Littlewood-Verall (L-V) モデル[2),3)]及び Littlewood のモデル[7)]の概要を述べる．

† Bayesian Statistical Models for Software Reliability by Hiroaki SANDOH (Department of Business Administration, Faculty of Commerce, University of Marketing & Distribution Sciences).

†† 流通科学大学商学部経営学科

2.1 ベイズ統計学の考え方

パラメータ θ をもつ確率分布を用いてなんらかのモデルを構築した場合を考える．そのモデルを実際に使用するためには，パラメータ θ の値を推定することが必要となる．パラメータの値をデータから統計的に推定する方法の代表的なものに，最ゆう推定法などがある．これらの推定方法は，パラメータ θ を確定変数として捉えるものであり，標本論の立場からの推定方法と呼ぶ．

これに対して，ベイズ統計学では θ を確率変数とみなして，その値に対する確信の度合いを確率分布を用いて表現する．さらに，このような考え方の下で導出された確率分布を用いてパラメータの値を推定したり，モデルそのものを直接評価したりする．パラメータを確率変数とみなすことの理由の詳細を，紙数をさいて説明することは本稿の目的から逸脱することになるのでここでは省略することとして，以下では，ベイズ統計学の考え方について概観する．なお説明の便宜上，連続型の確率分布に限定して考えることとする．

2.1.1 事前分布

未知のパラメータ θ をもつ確率分布の密度関数 (pdf) を $f(x|\theta)$ と書く．θ の値は未知ではあるが，分野によってはその道の専門家が「θ の値はおおむねこれくらいである」というような知識をもっていることも少なくない．このような情報を事前情報という．事前情報をもとにして，θ の値に関する確信の度合いを確率分布の形に表現したものを事前分布という．以下では，事前分布における確率密度（事前密度）を $p(\theta)$ と書く．事前分布の分散が十分に小さい場合や，実験を行って新たにデータを収集することが困難な場合には，事前分布を用いて θ の値を推定することも可能である．たとえば

$$\hat{\theta}=\int_{\theta}\theta p(\theta)d\theta, \qquad (1)$$

は事前平均であり，これを θ の推定量とすることもで

きる．あるいは，本来われわれが観測するのは確率変数 X の値であることから，X の値を次式を用いて予測することも可能である．

$$p(x)=\int_{\theta} f(x|\theta)p(\theta)d\theta. \qquad (2)$$

式(2)の左辺は確率変数Xの密度関数であるが，もはやパラメータ θ を含んでいないことから，パラメータを意識することなく，Xの振舞を予測することができる．

2.1.2 事後分布

たとえ θ の事前分布が定量化できたとしても，それだけでは情報が不十分な場合，新たに実験を行って確率変数 X の実現値をデータとして収集することで，θ に関する情報を更新することも可能である．

確率変数Xの実現値としてランダムサンプリングなどにより，n個の互いに独立なデータ $x_1, x_2, \cdots, x_n$ を入手した場合を考える．確率変数Xの本来の密度関数は $f(x|\theta)$ であることから，これらのデータ $x'=(x_1, x_2, \cdots, x_n)$ が得られるという同時確率密度は

$$f(x|\theta)=\prod_{i=1}^{n} f(x_i|\theta), \qquad (3)$$

となる．また，式(3)の同時確率密度をパラメータ θ の関数とみなした場合には，それをゆう度関数といい $L(\theta|x)$ のように書く，式(3)が得られると，データ x が得られた場合の θ の密度関数は，ベイズの定理により次式で与えられる．

$$p(\theta|x)=f(x|\theta)p(\theta)/\int_{\theta} f(x|\theta)p(\theta)d\theta. \qquad (4)$$

式(4)の右辺分母は，データ x が得られる確率を全ての可能な θ について求めたものである．一方，分子はパラメータが θ であるときにデータ x が得られる確率を表している．したがって，式(4)の右辺は，データ x が得られた場合，それがパラメータの値が θ であるときに得られたデータである確率を表している．式(4)左辺は，データ x を入手してから事後的に θ の確率を求めていることから事後密度と呼ばれる．事後密度が θ に関してもっている情報は，事前情報にデータ x のもつそれを加えたものとなっている．なお，式(4)右辺分母は単なる正規化定数とみなしてよい．

このようにして求めた事後分布を用いて θ に関する推定や，Xに関する予測などを行う場合には式(1)，(2)の事前密度を式(4)の事後密度に置き換えればよい．すなわち，次の式(5)，(6)を用いればよい．

$$\hat{\theta}=\int_{\theta} \theta p(\theta|x)d\theta, \qquad (5)$$

$$f(x|x)=\int_{\theta} f(x|\theta)p(\theta|x)d\theta, \qquad (6)$$

式(6)の左辺はパラメータ θ に依存しない X の確率密度を与えており，データ x が与えられた場合の予測密度と呼ばれる．

2.1.3 事前情報が少ない場合

これまで，事前情報に基づいて $p(\theta)$ を定量化した場合を前提としてきた．しかし，$p(\theta)$ を定量化するには事前情報が不十分であるような場合も少なくない．このような場合にも便宜上用いることのできる事前分布として何通りかが提案されている[1),19)]．ここでは，その主なものについて述べる．ただし，事前情報が不十分であることから，今後実験などによって，データを収集することを前提としていることに注意を要する．

一つは θ に関して何も分からないのなら，θ のどの値も同様に確からしいと考えて

$$p(\theta) \propto \text{const.}, \qquad (7)$$

とする方法である．式(7)のようにして得られる事前分布を局所一様事前分布という．もう一つは，事前分布として，式(4)で求められる事後分布が事前分布と同一の分布族に含まれるような確率分布を仮定するという方法である．たとえば，$f(x|\theta)$ が指数分布やガンマ分布の密度関数である場合には，θ の事前分布としてガンマ分布を選べば事後分布もガンマ分布となる．この性質は，数学的取扱い上非常に便利である．その理由は次のように説明できる．すなわち，事前分布と事後分布の違いは，ガンマ分布のパラメータが異なるだけであることから，事前分布であるガンマ分布のパラメータと事後分布のそれとの関係を求めておきさえすれば，式(4)のような複雑な計算をその都度行う必要がなくなるからである．このような考え方に基づく事前分布を共役事前分布という．

2.2 Littlewood-Verall モデル

Littlewood と Verall によって本モデルが提案される以前にもさまざまな信頼性に関するモデルが考案されている[20)~26)]．これらのモデルのほとんどは，次のように仮定している．

「デバッグを行った際には，確実にバグが検出・修正され，新たにバグが発生することはない．」

しかしながら，現実には，デバッグによって新たなバグが発生することが少なくない．L-V モデルはこのような問題点を緩和することを目的に考案されたモデルの一つである．

2.2.1 確率的に減少する故障率

Littlewood-Verall は，デバッグの際にソフトウェアに新しくバグが混入する可能性を，次のように捉えることによりモデル化した．

「デバッグ時に新しいバグが発生する可能性をも考えると，デバッグによりソフトウェアの故障率は確定的ではなく，確率的に減少する．」

このことは，第 $(i-1)$ 回目のソフトウェア障害と第 i 回目のそれとの時間間隔 X_i に対する故障率 λ_i を確率変数 Λ_i として取り扱うことにより，次式のように表現することができる．

$$\mathrm{Prob}[\Lambda_i<\lambda_0]\geqq\mathrm{Prob}[\Lambda_{i-1}<\lambda_0],\quad i=1,2,\cdots, \tag{8}$$

ここに，λ_0 は定数を表す．式(8)では，パラメータである故障率を確率変数として取り扱っているが，この段階では，まだベイズ統計学でいうところの事前分布・事後分布の考え方を用いているのではないことに注意する必要がある．

L-V モデルでは，Λ_i の振舞を与える確率分布として，密度関数が次式で与えられるガンマ分布 $\mathrm{Ga}(\alpha,\psi(i))$ を仮定している．

$$p(\lambda_i|\alpha)=p(\lambda|\alpha,\psi(i))=\frac{\psi(i)^\alpha\lambda^{\alpha-1}}{\Gamma(\alpha)}e^{-\psi(i)\lambda}. \tag{9}$$

ここに，$\Gamma(\cdot)$ はガンマ関数を表す．また，λ_i が式(8)の性質を満足するためには，$\psi(i)$ は i に関して増加関数でなければならない．

このように Λ_i の確率分布として式(9)のガンマ分布を仮定する理由としては，次のことがあげられる．

- 得られる結果が数学的に取り扱いやすいこと
- ガンマ分布の密度関数はパラメータ α の値に応じてさまざまな形状をとることができるという意味で柔軟性に富んでいること

ここで，他の信頼性に関するモデルと同様，ソフトウェア障害時間間隔 X_i の確率密度関数として

$$f(x_i|\lambda_i)=\lambda_i\exp(-\lambda_i x_i), \tag{10}$$

を考えると，λ_i の振舞を表す確率分布として式(9)を仮定したことから，次式が得られる．

$$f(x_i|\alpha,\psi(i))=\int_0^\infty f(x_i|\lambda_i)p(\lambda_i|\alpha)d\lambda_i=\frac{\alpha\psi(i)^\alpha}{[x_i+\psi(i)]^{\alpha+1}}. \tag{11}$$

式(11)は，パレート分布の密度関数であり，未知のパラメータとして $\alpha,\psi(i)$ を含んでいる．$\psi(i)$ の構造及びその中に含まれるパラメータの推定方法については **3.** で述べることとし，今後は $\psi(i)$ を既知として考える．したがって，以下では α のみに注目し，α に対してベイズ統計学の考え方を適用する．

2.2.2 α の事後分布

現時点までに n 回のソフトウェア障害が発生しており，障害時間間隔データ $\boldsymbol{x}'=(x_1,x_2,\cdots,x_n)$ を入手しているものとする．このようなデータ $\boldsymbol{x}$ が得られる確率を α の条件付確率として求めると次式を得る．

$$f(\boldsymbol{x}|\alpha)=\int_0^\infty\cdots\int_0^\infty\prod_{i=1}^n f(x_i|\lambda_i)p(\lambda_i|\alpha)d\lambda_i=\prod_{i=1}^n\left[\frac{\alpha\psi(i)^\alpha}{(x_i+\psi(i))^{\alpha+1}}\right]. \tag{12}$$

ここで，α に関して事前になんら情報がない状況を想定し，事前分布として **2.1.3** に述べた局所一様事前分布を仮定する．すなわち

$$p(\alpha)\propto\mathrm{const.}, \tag{13}$$

を仮定する．このとき，α に関する事後密度はゆう度関数が式(12)のように求められたことから，式(4)に式(12)，(13)を代入して必要な計算を行うことにより

$$p(\alpha|\boldsymbol{x})=\frac{\eta^{n+1}\alpha^n}{\Gamma(n+1)}\exp(-\eta\alpha), \tag{14}$$

となる．なお，式(14)はガンマ分布 $\mathrm{Ga}(n+1,\eta)$ の密度関数となっている．ここに

$$\eta=\log\theta, \tag{15}$$

$$\theta=\prod_{i=1}^n\theta_i, \tag{16}$$

$$\theta_i=(x_i+\psi(i))/\psi(i), \tag{17}$$

である．

Littlewood-Verall は，ソフトウェアの信頼性を評価する際には，上のパラメータ α の値を推定することよりも，次のソフトウェア障害あるいは k 回先のソフトウェア障害がいつ起こるのかを予測することのほうが重要であると考え，これらの確率変数の予測分布を導出した．以下では，これらの確率変数の振舞を予測する方法について概説する．

2.2.3 X_{n+1}, X_{n+k} の予測

確率変数 X_{n+1} は，第 n 回目のソフトウェア障害が生起し，そのデバッグが完了した時点から計測して，次のソフトウェア障害が起こるまでの時間を表すものとする．この確率変数の振舞を表す本来の確率分布は故障率 λ_{n+1} の指数分布である．なお，指数分布の密度関数は式(10)に示すとおりである．一方，λ_{n+1} の

振舞は，式(9)において i を $n+1$ に置き換えたガンマ分布で規定される．さらに，式(9)のガンマ分布のパラメータ α の，現時点における事後分布は式(14)で与えられる．これらのことから，確率変数 X_{n+1} の予測分布は

$$f(x_{n+1})\equiv f(x_{n+1}|\boldsymbol{x})$$
$$=\int_0^\infty f(x_{n+1}|\lambda_{n+1})p(\lambda_{n+1}|\alpha)(p(\alpha|\boldsymbol{x})d\alpha$$
$$=\frac{(n+1)\eta^{n+1}}{x_{n+1}+\psi(n+1)}[\eta+\log\theta_{n+1}]^{-(n+2)}, \tag{18}$$

となる．ただし，θ_{n+1} は x_{n+1} の関数である．

以上のようにして導出した X_{n+1} に対する予測分布を用いれば，次に起こるソフトウェア障害に対する信頼度やパーセント点を容易に求めることができる．ただし，X_{n+1} に関する期待値は積分が発散するため，存在しない．次のソフトウェア障害の時間 t における信頼度は次式のようになる．

$$R(t)=\left[\frac{\eta}{\eta+\log\zeta(t)}\right]^{n+1}, \tag{19}$$

だたし

$$\zeta(t)=\frac{t+\psi(n+1)}{\psi(n+1)}, \tag{20}$$

である．さらに，時刻 t における故障率を計算すると

$$z(t)=\frac{n+1}{[t+\psi(n+1)][\eta+\log\zeta(t)]}, \tag{21}$$

が得られる．また，信頼度 $100q\%$ をもつパーセント点 $y_{q,n+1}$ は次式となる．

$$y_{q,n+1}=\psi(n+1)\{\theta^{[(1-q)^{-1/(n+1)}-1]}-1\}. \tag{22}$$

現在からさらに k 回先のソフトウェア障害時間間隔を確率変数 X_{n+k} で表すとき，その振舞の予測は，上での導出結果において $n+1$ とした箇所を $n+k$ とすればよい．なお，現在時点が第 n 回目のソフトウェア障害が発生してからすでに何時間が経過している時点である場合にも，上と同様な議論も可能であるが，ここでは省略する．

2.3 Littlewood のモデル

一般に，ソフトウェア障害発生時にデバッグを行うことにより，ソフトウェアの故障率は減少する．しかし，その減少する大きさは，多くのモデルのように確定的ではなく，確率的な振舞を示すと考えるほうがより自然である．Littlewood はこのような性質をもつモデルの開発を試みた．

2.3.1 各バグとソフトウェアの故障率

Littlewood のモデルでは，次のように仮定する．

- ソフトウェアは初期状態において N 個のバグを含んでいる．
- ソフトウェア障害が生起するたびに，デバッグを実施し，その都度バグは一つずつ検出・修正される．
- N 個のバグのうち，おのおののバグが原因となってソフトウェア障害が起きる故障率は $\phi_i(i=1, 2, \cdots, N)$ である．換言すれば，ソフトウェアのテストを開始してから，それぞれのバグがソフトウェア障害を引き起こすまでの時間は，故障率 ϕ_i の指数分布に従う．

これらの仮定は，同時に次のことをも意味する．それは，第 i 回目のソフトウェア障害の時間間隔 X_i は，故障率が次式で与えられる指数分布に従うということである．

$$\lambda_i=\phi_1+\phi_2+\cdots+\phi_{N-i+1}. \tag{23}$$

ここで，デバッグによって減少する故障率の大きさが確率的な振舞を示すことをモデルとして表現するため，各バグに対する故障率 ϕ_i を確率変数 Φ_i に置き換えて考える．したがって，式(23)のソフトウェアの故障率も確率変数として大文字に置き換えて考える．

ここで，Φ_i は互いに独立にガンマ分布 $\mathrm{Ga}(\alpha, \beta)$ に従うものと仮定する．ただし，ここでの仮定は，L-V モデルとは異なり，ベイズ統計学でいうところの事前分布としての仮定である．なお，事前分布としてガンマ分布を用いたのは，**2.1.3** に述べた共役事前分布の考え方に基づいている．以下では，N, α, β の値が既知であるとして話を進める．なお，これらのパラメータの推定方法は次章に述べる．

2.3.2 ソフトウェア故障率に対する事後分布

初期状態においてソフトウェアに含まれていた N 個のバグを 1，2，…，N で表し，現在までに n 個のバグ $N-n+1, \cdots, N$ が検出・修正されているという状況を想定する．ただし，ソフトウェアのテストが開始されてから現在まで τ 時間が経過しているものとする．以下では，このような事象を E_0 と書くこととする．

現時点でソフトウェアに残存しているバグのうちバグ $k(k=1, 2, \cdots, N-n)$ の故障率 φ_k について考える．事象 E_0 が生起したことは，同時に，バグ k が τ 時間の間ソフトウェア障害を起こさなかったということを意味する．バグ k に関するこのような事象を E_k と書くこととすると，Φ_k の事後密度は，ベイズの定理により

$$p(\phi_k|E_k)=c\cdot p(\phi_k|\alpha, \beta)\,\mathrm{Prob}\,[E_k|\phi_k], \tag{24}$$

で与えられる．ここに，$p(\phi_k|\alpha,\beta)$ は Φ_k の事前密度を表しており，ガンマ分布 $\mathrm{Ga}(\alpha,\beta)$ の密度関数である．また，c は正規化定数を表す．一方，指数分布の仮定から

$$\mathrm{Prob}[E_k|\phi_k]=\exp(-\phi_k\tau), \tag{25}$$

が成立する．これらのことを考慮して，式(24)を計算すると，その結果は $\mathrm{Ga}(\alpha,\beta+\tau)$ の密度関数となることが容易に示される．

ソフトウェアの現時点での故障率 Λ_{n+1} は $\Phi_k(k=1,2,\cdots,N-n)$ の和で与えられる．Φ_k は互いに独立であり，事後分布において $\mathrm{Ga}(\alpha,\beta+\tau)$ に従うことから，Λ_{n+1} に対する事後分布は $\mathrm{Ga}[(N-n)\alpha,\beta+\tau]$ となる．なお，今後 Λ_{n+1} の事後密度を $p(\lambda_{n+1}|E_0)$ と書くこととする．

2.3.3 予測分布

ソフトウェアのテストが開始されてから，現在までに τ 時間が経過しており，これまで，n 回のソフトウェア障害が発生している．現在から計測して，次のソフトウェア障害が生起するまでの時間を確率変数 X で表す．第 n 回目のソフトウェア障害が生起したことにより故障率が Λ_n から Λ_{n+1} に変化した時点から，現在までにすでに何時間かが経過しているが，無記憶性をもつ指数分布を用いていることから，確率変数 X は本来の故障率 Λ_{n+1} の指数分布に従うことが分かる．

2.3.2 でみたように，現時点でのソフトウェアの故障率 Λ_{n+1} は，事後分布において $\mathrm{Ga}[(N-n)\alpha,\beta+\tau]$ に従うことから，X の予測密度は次のようにして導出される．

$$\begin{aligned} f(x)&\equiv f(x|E_0)\\ &=\int_0^\infty f(x|\lambda_{n+1})p(\lambda_{n+1}|E_0)d\lambda_{n+1}\\ &=\frac{(N-n)\alpha(\beta+\tau)^{(N-n)\alpha}}{(\beta+\tau+x)^{(N-n)\alpha+1}}, \end{aligned} \tag{26}$$

ここで，式(26)もパレート分布の密度関数になっていることが分かる．

確率変数 X の予測密度が導出できたことから，次のソフトウェア障害の時間 t における信頼度及び時刻 t における故障率は，それぞれ式(27)，(28)となる．

$$R(t)=\left[\frac{\beta+\tau}{\beta+\tau+t}\right]^{(N-n)\alpha}, \tag{27}$$

$$z(t)=(N-n)\alpha/(\beta+\tau+t). \tag{28}$$

同様に，式(26)を用いて確率変数 X の期待値を計算すると，$(N-n)\alpha>1$ かつ $\alpha\geqq 1$ が成立する場合に

$$E(X)=(\beta+\tau)/[(N-n)\alpha-1], \tag{29}$$

が得られる．

以上，次のソフトウェア障害が生起するまでの時間に関する予測分布を中心に展開した．Littlewood のモデルを用いた場合，上述のような議論ばかりでなく，これまでと同じデバッグプロセスを継続することを前提として次の項目についての議論も可能である．

- 現在から計測して，さらに k 回ソフトウェア障害が生起するまでの時間に関する予測
- 現在から計測して，さらに k 回ソフトウェア障害が生起した時点での信頼性予測
- 現在からさらに τ' 時間経過後の信頼性予測

なお，これらについては，紙数の関係上省略する．

3. パラメータの推定に関する問題

2. では，L-V モデル，Littlewood のモデルを概観した．ここでは，これらのモデルにおけるパラメータ推定に関する問題について考察し，その推定方法についても言及する．

3.1 L-V モデル

2.2 では，パラメータ $\psi(i)$ を既知として L-V モデルを展開した．ここでは，この $\psi(i)$ の構造ならびに $\psi(i)$ に含まれるパラメータの推定方法に関する問題について述べる．

前にも述べたように，L-V モデルがソフトウェアの信頼性向上の様子を表現できるためには，$\psi(i)$ は i に関して増加関数でなければならない．$\psi(i)$ の構造は，基本的には本モデルのユーザが自由に決定すればよいのであるが，さまざまな構造を対象として考える場合には，その中で最も適切な構造を決定することが必要となる．この問題に対しては，Littlewood-Verall は Cramer-von Mises の統計量に基づく適合度検定を用いて最適な構造を決定している[2]．ただし，この方法を採用する場合には，膨大な計算量を必要とするなどの難点も存在する．

また，Abdel-Ghaly, Chan & Littlewood[12] は，$\psi(i)$ の構造として最も単純な次式を仮定し，他のモデルとの比較をも行っている．

$$\psi(i)\equiv\psi(i,\boldsymbol{\beta})=\beta_1+\beta_2 i(\beta_2>0). \tag{30}$$

$\psi(i)$ の構造を固定して考えた場合（たとえば式(30)），パラメータ $\boldsymbol{\beta}'=(\beta_1,\beta_2)$ の推定が次なる問題として生じるが，パラメータ推定は以下のようにして行うことができる．

2.2 と同様，現在までに n 回のソフトウェア障害が観測されており，n 個のソフトウェア障害時間間隔

データ $\boldsymbol{x}'=(x_1, x_2, \cdots, x_n)$ を入手しているものとする．このとき，これらのデータに対するゆう度関数が式(12)のようになることはすでに述べたとおりである．しかし，式(12)を用いて $\boldsymbol{\beta}'=(\beta_1, \beta_2)$ を推定するには次のような問題点が残されている．すなわち，式(12)はパラメータ α を含んだ形である．L-V モデルでは α に対する取扱いにベイズ論的アプローチを採用していることから，α を残したままの式(12)を用いて最ゆう推定はできない（α に対してベイズ論的アプローチを採るのではなく，これに対しても最ゆう推定を行う場合には式(12)を用いることが可能である）．

そこで，見方を変えて **2.2.3** で導出した予測分布に目を向ける．式(18)で与えられる予測密度はパラメータとして過去のデータ $\boldsymbol{x}$ と $\phi(i)$ を含んでいるものの，もはや α は含まれていない．この意味で，この予測密度を用いた最ゆう推定は可能である．その方法は，以下のとおりである．

式(18)より，データ $(x_1, x_2, \cdots, x_{n-1}, x_n)$ を入手している場合の X_n に対する予測密度は

$$f(x_n|x_1, x_2, \cdots, x_{n-1}) = f(x_n|E_{n-1}) = \frac{n\eta_n^n}{x_n+\phi(n)}[\eta_n+\log\theta_n]^{-(n+1)}, \quad (31)$$

と書くことができる．ただし，E_{n-1} は時間 $\sum_{i=1}^{n-1} x_i$ までに $(n-1)$ 個のソフトウェア障害時間間隔データ $(x_1, x_2, \cdots, x_{n-1})$ が得られたという事象を表す．また，η_n の定義は

$$\eta_n=\sum_{i=1}^{n-1}\log\theta_i, \quad (32)$$

であり，θ_i のそれは式(17)のとおりである．したがって，現在までに $\boldsymbol{x}'=(x_1, x_2, \cdots, x_{n-1}, x_n)$ が得られたという事象に対するゆう度関数は

$$L(\beta_1, \beta_2)=\prod_{i=1}^{n} f(x_i|x_1, x_2, \cdots, x_{i-1}) = \prod_{i=1}^{n}\frac{i\eta_i^i}{x_i+\phi(i)}[\eta_i+\log\theta_i]^{-(i+1)}, \quad (33)$$

となる．よって，式(33)を最大にするような β_1, β_2 を求めればよい．

式(33)をみても分かるように，以上の最ゆう推定自身にも相当量の計算が必要である．前述したように，$\phi(i)$ の構造を決定する際にも膨大な計算を必要とすることから，L-V モデルを緩和し，計算量を比較的少なくする方法も提案されている[12]．

3.2 Littlewood のモデル

本モデルで推定すべきパラメータは初期状態でのバグ数 N と事前分布のパラメータ α, β である．これらのパラメータの値を推定する場合にも，**3.1** に述べた方法と同様，予測分布に基づく最ゆう推定法を用いればよい．すなわち，時間間隔データ $\boldsymbol{x}'=(x_1, x_2, \cdots, x_n)$ を入手している場合のゆう度関数を式(26)の予測密度を用いて導出すると

$$L(N, \alpha, \beta) = \prod_{i=1}^{n}\frac{(N-i+1)\alpha\left(\beta+\sum_{j=1}^{i-1}x_j\right)^{(N-i+1)\alpha}}{\left(\beta+\sum_{j=1}^{i-1}x_j+x_i\right)^{(N-i+1)\alpha+1}}, \quad (34)$$

となる．よって，式(34)を最大にするような N, α, β を求めればよい．

4. 事 例

これまでに概観した Littlewood らのモデルは，ソフトウェア中の残存バグ数ではなく，次のソフトウェア障害がいつ生起するのかを予測することに焦点を絞っている．ここでも，同様な観点からのモデルの適用例を示すこととし，Jelinski-Moranda (J-M) モデル[28]を適用した場合との比較を行う．

ここで，Jelinski-Moranda (J-M) モデルとは，i 回目のソフトウェア障害時間間隔 X_i に対する確率密度関数を次式で与えるモデルである．

$$f(x_i|\lambda_i)=\lambda_i\exp(-\lambda_i x). \quad (35)$$

ただし

$$\lambda_i=\phi(N-i+1), \quad i=1, 2, \cdots, N, \quad (36)$$

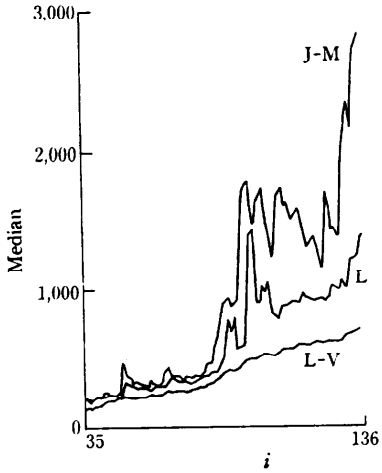
であり，Nは初期状態におけるソフトウェア中の総バグ数，ϕは各バグの故障率を表す．本モデルにおけるパラメータはN及びϕであるが，これらの値は，ソフトウェア障害時間間隔データ $\boldsymbol{x}'=(x_1, x_2, \cdots, x_n)$ により，最ゆう推定法を用いて推定する．

表-1 に，解析対象としたソフトウェア障害時間間隔データを示す．なお，これは文献27) に掲載されたデータである．**図-1** に J-M モデル，L-V モデル及び Littlewood のモデルを適用した結果を示す．これは，現在 $(x_1, x_2, \cdots, x_{i-1})$ なるデータが存在するものとした場合に，次のソフトウェア障害までの時間 X_i のメジアンを求めた結果である．なお，メジアンを用いたのは，**2.2.3** に述べたように，L-V モデルでは，予測分布の期待値が存在しないためである．

図-1 から分かるように，J-M モデルはデータの一つ一つに影響され，メジアンの推定値が大きく変動

表-1 ソフトウェア障害時間間隔データ

3.	30.	113.	81.	115.	9.	2.
91.	112.	15.	138.	50.	77.	24.
108.	83.	670.	120.	26.	114.	325.
55.	242.	68.	422.	180.	10.	1146.
600.	15.	36.	4.	0.	8.	227.
65.	176.	58.	457.	300.	97.	263.
452.	255.	197.	193.	6.	79.	816.
1351.	148.	21.	233.	134.	357.	193.
236.	31.	369.	748.	0.	232.	330.
365.	1222.	543.	10.	16.	529.	379.
44.	129.	810.	290.	300.	529.	281.
160.	828.	1011.	445.	296.	1755.	1064.
1783.	860.	983.	707.	33.	868.	724.
2323.	2930.	1461.	843.	12.	261.	1800.
865.	1435.	30.	143.	108.	0.	3110.
1247.	943.	700.	875.	245.	729.	1897.
447.	386.	446.	122.	990.	948.	1082.
22.	75.	482.	5509.	100.	10.	1071.
371.	790.	6150.	3321.	1045.	648.	5485.
1160.	1864.	4116.				



L-V : Littlewood-Verall model
L : Littlewood model
J-M : Jelinski-Moranda model

図-1 モデルの比較

している．これに対し，Littlewood のモデルはやや安定，L-V モデルでは非常に安定した推定値を提供している．さらに，L-V モデルの大きな特徴として，推定値が安定しているばかりでなく，かなり控え目な値を示すということがあげられる．換言すれば，L-V モデルを用いて得られた推定値は，他のモデルに比べ安全側のものである可能性が大である．

5. おわりに

本稿では，ソフトウェアの信頼性を定量的に評価するための種々のモデルのうち，ベイズ統計学の立場から考案されたモデルのいくつかを紹介した．ここでは，Littlewood-Verall (L-V) モデルと Littlewood のモデルを紹介するに留めた．前者は，現実のソフトウェアのデバッグは完全ではないということの表現を試みたものである．後者は，デバッグによってバグが確実に修正されたとしても，減少する故障率の大きさは確定的ではないということを表現しようとしたものである．ただし，これらのモデルをそのまま実際に適用する場合には，相当量の計算が必要であるのも事実である．このためか，我が国ではこれらのモデルを適用した例はほとど報告されていない．本文中にも述べたように，これらのモデルを緩和して計算量を少なくする方法も存在することから，我が国においても一度適用してみることが必要であろう．

参考文献

1) Marz, H. F. and Waller, R. A.: Bayesian Reliability Analysis, John Wiley, New York (1982).
2) Littlewood, B. and Verall, J. V.: A Bayesian Reliability Growth Model for Computer Software, J. Roy. Statist. Soc., C, Vol. 22, No. 3, pp. 332-346 (1973).
3) Littlewood, B. and Verall, J. V.: A Bayesian Reliability Model with a Stochastically Monotone Failure Rate, IEEE Trans. Reliab., Vol. R-23, No. 4, pp. 108-114 (1974).
4) Smith, A. F.: A Bayesian Note on Reliability Growth during a Development Testing Program, IEEE Trans. Reliab., Vol. R-26, No. 5, pp. 346-347 (1977).
5) Littlewood, B.: How to Measure Software Reliability and How not to, IEEE Trans. Reliab., Vol. R-28, No. 2, pp. 103-110 (1979).
6) Littlewood, B.: Theories of Software Reliability, IEEE Trans. Software Eng., Vol. SE-6, No. 5, pp. 489-500 (1980).
7) Littlewood, B.: Stochastic Reliability Growth —A Model for Fault-Removal in Computer Programs and Hardware Designs, IEEE Trans. Reliab., Vol. R-30, No 2, pp. 313-320 (1981).
8) Keiller, P. A. et al.: Comparison of Software Reliability Prediction, Dig. FTCS 13, pp. 128-134 (1983).
9) Jewell, W. S.: Bayesian Estimation of Undetected Errors, Theory of Reliablity: Proc. of

Internl. Physics, Serra, A. and Barlow, R. E., eds., North-Holland, pp. 405-425 (1984).
10) Catuneanu, V. M. and Mihalache, A. N.: Improving Accuracy of the Littlewood-Verall Model, IEEE Trans. Reliab., Vol. R-34, No. 5, pp. 418-421 (1985).
11) Jewell, W. S.: Bayesian Extentions to a Basic Model of Software Reliability, IEEE Trans. Software Eng., Vol. SE-11, No. 12, pp. 1465-1471 (1985).
12) Abdel-Ghaly, A. A., Chan, P. Y. and Littlewood, B.: Evaluation of Competing Software Reliability, IEEE Trans. Software Eng., Vol. SE-12, No. 9, pp. 950-967 (1986).
13) Fard, N. S. and Dietrich, D. L.: A Bayesian Reliability Growth Model for a Development Testing Program, IEEE Trans. Reliab., Vol. R-36, No. 5, pp. 568-571 (1987).
14) Bai, D. S. and Yun, W. Y.: Optimum Number of Errors before Releasing a Software System, IEEE Trans. Reliab., Vol. R-37, No. 1, pp. 41-44 (1988).
15) 三道弘明, 藤井 進, 山脇達也：離散型ソフトウェア信頼性解析に対するベイズ論的1人デバッガモデルとその応用, 電子情報通信学会論文誌, Vol. J 72-A, No. 3, pp. 590-596 (1989).
16) 三道弘明：離散型ソフトウェアの疑似 Error-Seeding によるテストに対する最適リリース問題, 電子情報通信学会論文誌, Vol. J 72-A, No. 6, pp. 992-994 (1989).
17) 三道弘明, 山脇達也, 藤井 進：離散型ソフトウェアの信頼性解析に対するベイズ論的2人デバッガモデルとその応用, 電子情報通信学会論文誌, Vol. J 72-A, No. 7, pp. 1110-1116 (1989).
18) 三道弘明, 澤田 清：ソフトウェアに対するゼロ障害型信頼性実証試験に関する研究, 電子情報通信学会論文誌, Vol. J 73-A, No. 3, pp. 564-569 (1990).
19) Box, G. E. P. and Tiao, G. C.: Bayesian Inference in Statistical Analysis, Addison Wesley, Massachusetts (1973).
20) Schick, G. J. and Wolverton, R. W.: An Analysis of Competing Software Reliability Models, IEEE Trans. Software Eng., Vol. SE-4, No. 2, pp. 104-120 (1978).
21) Ramamoorthy, C. V. and Bastani, F. B.: Software Reliability—Status and Perspectives, IEEE Trans. Software Eng., Vol. SE-8, No. 4, pp.354-371 (1982).
22) Shantikumar, J. G.: Software Reliability Models—A Review, Microelectronics & Reliability, Vol. 23, No. 5, pp. 903-943 (1983).
23) Goel, A. L. and Bastani, F. B., eds.: Special Issue on Software Reliability, IEEE Trans. Software Eng., Vol. SE-11, No. 12, pp. 1409-1517 (1985).
24) Goel, A. L. and Bastani, F. B., eds.: Special Issue on Software Reliability—Part II, IEEE Trans. Software Eng., Vol. SE-12, No. 1, pp. 1-181 (1986).
25) Mellor, P: Software Reliability Modeling—The State of the Art, Information and Software Technology, Vol. 29, No. 2, pp. 81-98 (1987).
26) 三道弘明：ソフトウェアの定量的信頼性評価法, システム/制御/情報, Vol. 34, No. 6, pp. 316-323 (1990).
27) Musa, J. D.: Software Reliability Data, Data and Analysis Center for Software, Rome Air Development Center, Rome, NY, Tech. Rep.
28) Jelinski, Z. and Moranda, P. B.: Software Reliability Research, Statistical Computer Performance Evaluation, Freiberger, W., ed. New York, Academic, pp. 465-484 (1972).

(平成2年9月3日受付)