ソフトウェアデバッグシミュレーションモデルにおける制御パラメータのキャリブレーションに関する研究

中嶋, 浩人 / NAKAJIMA, Hiroto

(発行年 / Year)
2008-03-24
(学位授与年月日 / Date of Granted)
2008-03-24
(学位名 / Degree Name)
修士(工学)
(学位授与機関 / Degree Grantor)
法政大学 (Hosei University)

要　旨

本研究では，ソフトウェア信頼性評価のための時系列データを生成するモンテカルロシミュレーションモデルと，それに含まれる，制御パラメータのキャリブレーションの方法について考察する．この制御パラメータの最適な値は実際に観測されたソフトウェアフォールト発生事象に関するデータとシミュレーションによって生成されたデータの間の誤差二乗和を平均的に最小化するものして得られる．これにより，本研究で提案されたキャリブレーション法を用いることによって，モンテカルロシミュレーションモデルを一般の確率過程モデルよりも緩い仮定をもつソフトウェア信頼性評価手法として利用することが可能となる．また，いくつかの数値例を示し，シミュレーションモデルを用いたソフトウェア信頼性評価例と，今後の課題について述べる．

Abstract

In this study, we try to develop a calibration method of control parameters included in three Monte Carlo simulation models which generate time series data for software reliability assessment. The control parameters are calibrated so as to minimize the mean sum of squared errors between the actually observed data and the generated ones. As a result, the proposed method enables us to use the simulation model for assessing the software reliability based on the observed data. We show several numerical examples of the calibration and software reliability assessment by the simulation models.

目 次

1. はじめに

ソフトウェアの定量的な信頼性評価を主な目的として採取された，テスト時刻とそこまでに発見･修正されたソフトウェアフォールト数のいくつかの組で表されるような，時系列データに基づいたソフトウェア信頼度成長モデルにおいては，従来より数多くの確率過程モデルが提案されてきている[1,2,3]. これらの確率過程モデルを用いることの利点の一つは，ソフトウェア信頼度，MTBFなどといった確率則に基づいたいくつかの信頼性評価尺度が得られることにある．しかしながら，これらのモデルは一般に厳しい仮定の下で成立するモデルであるとも言える．例えば，非同次ポアソン過程(NHPP)モデルはソフトウェア信頼度評価モデルとして広く知られているが，これらは実際のソフトウェアのテスト工程において自然に発生することが知られている，複数のソフトウェアフォールトが同時に発見されるなどといった現象の発生を，厳密には記述することができない．過去の研究において，このことに対する一つのアプローチとして，Sahinoglu[4]は複合ポアソン過程モデルを提案した．しかしながら，複合ポアソン過程モデルに含まれる未知パラメータは実測されたデータを用いた統計的推定が難しいということも知られており，モデルの実用性という点においては難点がある．したがって，数理モデルがその成立のために必要とした仮定が現実的ではないと考えられる場合には，別のアプローチを試みることも有用かもしれない．

もう少し一般的に説明する．いま，時系列データ$(t_i, d_i)(i = 1, 2, 3, \ldots, I)$を，ある確率的な振舞いをするシステムから観測されるものと仮定する．ここで，d_iはシステムの時刻がのときのシステムの出力結果である．このようなデータを扱うために，多くの場合d_iは時刻t_iで得られるある確率過程の実現値と見なされる．したがって，もしそのデータセットに対して適切かつ取り扱いやすい確率過程モデルを見出すことが出来れば，そのモデルに含まれる未知パラメータを推定する

ことや，それに基づいていくつかの確率的特性を考慮した将来の値$(t_i, d_i)(i = I + 1, I + 2, \ldots)$を比較的容易に予測できるかもしれない．しかしながら，このような確率モデルのいくつかのものは，何らかの非現実的な，数学的な解析の容易さのための厳しい仮定を必要としていることがある．これに対して，シミュレーションによるモデルはある程度の柔軟性を持っている．すなわち，シミュレーションモデルは一般に確率システムを表現するモデルとしては比較的複雑で定式化が困難と思われる現象をもアルゴリズムとして表現し，コンピュータ上に実装することが出来る．しかし，このようなシミュレーションモデルから得られる結果は，前もって決められたモデルの制御パラメータの値によって左右されることから，通常我々は一方向的にシミュレーションの実行結果を取得し，制御パラメータを摂動させることによる感度分析を行うのが通例となっている．

そこで本研究では，最小二乗法に基づいた，モンテカルロシミュレーションモデルの制御パラメータのキャリブレーションの方法について提案・考察する．すなわち，実際に得られた時系列データを再現するようなシミュレーションモデルの制御パラメータを推定する方法について議論する．ここで，具体的なシミュレーションモデルとして，ソフトウェアのテスト工程において発生するソフトウェア故障に関する時系列データの振舞いを記述するモデルを構築し，制御パラメータのキャリブレーションの方法を提案する．また，ソフトウェアの信頼性評価尺度の推定値をシミュレーションモデルを経由して得ることを試み，数値例として示す.

2. シミュレーションモデル

本節では以下に使用するキャリブレーションモデルや，シミュレータについて説明していく. 本研究では３つのモデルについて考察，研究を行う.

2.1 1パラメータモデル

ソフトウェアフォールトデバッグ工程において，簡単なブラックボックステストを考える. 図１のように，ある入力が正しいプログラム論理を通って出力される際，正しい結果として出力され，誤りを含んだプログラム論理を通って出力される際，誤った結果として出力される. その際，テストする順番はランダムであるとする. 1パラメータモデルでは，要求仕様に照らして期待通りでない結果（以下では単に誤った結果と呼ぶことにする）が出力された際は，これが毎回確実に修正され，正しい結果が出力されるように修正が行われると仮定する. この際，シミュレーションにおける制御パラメータとして設定されるものは，プログラム上のソフトウェアフォールト混入率である.

図 1: 入力・プログラム・出力の関係の概念図.

2.1.1 シミュレーションアルゴリズム(1パラメータ)

ソフトウェアのテスト工程においては，以下に述べる簡単なブラックボックステストが行われたものと仮定する. そのテストのシミュレーションアルゴリズムを以下に示す. また，以下の記号を定義する.

S_k $(k = 1, 2, 3, \ldots, K)$: テスト工程におけるk番目のソフトウェアフォールト発見時刻の実現値

K: 検出されたソフトウェアフォールト数

ここで，S_kは実際に得られている実測データである．

Step 1: テストされるプログラムコードかに見立てた1次元配列変数p[1...psize]を設定する．ここでpsizeは任意の自然数で与えられる．またこれはソフトウェアシステムのプログラムサイズに関係する値である．つまり，プログラム全体をテストされるプログラム論理ごとにpsize個のセルに分割すると考えてよい．

Step 2: $z_i = 0, i = 1, 2, 3, \ldots, S_k$とする．また，$I = S_k$が成り立つ．

Step 3: $l = 1$から$l =$ psizeまで動かし以下を定める．

$$p[l] = \begin{cases} 1 \text{ with probability } \theta_1 \\ 0 \text{ with probability } 1 - \theta_1 \end{cases}, \tag{2.1}$$

ここで，値‘1’はそのプログラムコードがソフトウェアフォールトを含んでいることを意味し，‘0’はフォールトフリーなコードであることを意味する．また，θ_1はプログラムコード当りの不信頼度を意味する．つまり，θ_1の確率で，pisize個のセルの中にフォールトが含まれているということである．

Step 4: $I = 1$とする．

Step 5: 区間[1, psize]から自然数cをランダムに1つ選ぶ．もし

p[c] = 1 なら $z_i = z_i + 1$ とし，p[c] = 1 とする．

Step 6: $t_i < S_k$ のときに $i = i + 1$ としStep 5を行う. それが満たされなくなったときStep 7を行う.

Step 7: f_1 の値を返す. すなわち $f(t_i|\Theta) = z_i (i = 1, 2, 3, \ldots, S_k)$ となる．ここでΘは制御パラメータからなるベクトルであり，$f(t_i|\Theta)$は制御パラメータベクトルΘが所与の下で，時刻t_iにおけるシミュレーションプログラムの出力結果を表す.

上記の結果，シミュレートされたフォールト発見時刻のデータ $(t_i, f(t_i|\Theta))(i = 1, 2, 3, \ldots, S_k)$ を得る．図2にf_1のサンプルパスを示す. ここではpsize=500, および $\theta_1 = 0.15$ とした．この場合，$\Theta = \theta_1$ であることに注意が必要である．

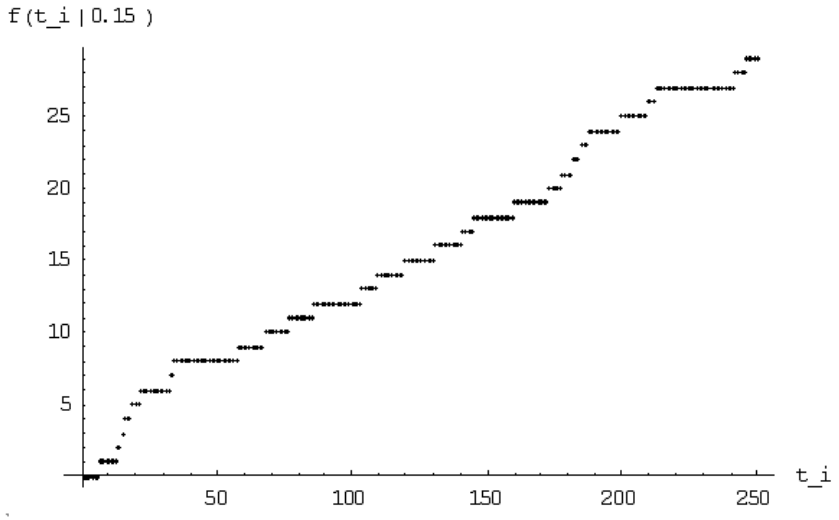

図 2: 生成されたシミュレーションデータ.

2.2 2パラメータモデル(不完全デバッグを考慮したモデル)

前節のモデルを拡張してソフトウェアフォールトデバッグ工程における,不完全性を考慮したモデルを考える.つまり,誤ったプログラム論理が発見され,その原因となっていたソフトウェアフォールトが修正される際に,そのフォールトは修正されるが,他の部分に対して悪影響を与え,新たなフォールトを埋め込む可能性を考える.これはいわゆる不完全デバッグモデルであり,従来より多くの確率モデルが提案されてきている.

2.2.1 シミュレーションアルゴリズム(2パラメータ)

一般に不完全デバッグモデルはモデルパラメータを推定することが難しいということも知られている.

一方,我々のモデルにおいては,1パラメータシミュレータのStep5を以下の5'に変更することにより簡単な不完全デバッグモデルを得ることができる.

> Step 5 : 区間$[1, \mathrm{psize}]$の中からランダムに整数cを選ぶ.もし$\mathrm{p}[c] = 1$ならば$z_i = z_i + 1$とし,$1 - \theta_2$の確率で$\mathrm{p}[c] = 0$に変更する.もし$\mathrm{p}[c] = 0$ならばStep 6に進む$(0 \leq \theta_2 \leq 1)$.

この場合,2つの制御パラメータθ_1およびθ_2についてキャリブレーションを行うことを考える.θ_1はプログラムコード当りの不信頼度であり,θ_2はデバッグにおける不完全デバッグ率である.ここでテストによって発見されたソフトウェアフォールトは除去され,その記録は残るが,実際には新たなフォールトがそのプログラムコード中に埋め込まれてしまったことを意味する.

2.3 3パラメータモデル(不確実デバッグを考慮したモデル)

ここでは，1パラメータモデル，2パラメータモデルをさらに拡張し，3パラメータモデルを構築する．ここでは特に，テスト工程におけるランダム性とテスト工程の質との関係をモデルに盛り込むことを考える．

前節までに述べたように，テスト工程においてブラックボックステストを想定し，本研究での1パラメータモデル，2パラメータモデルの仮定を設定するならば，それは質の悪いテスト工程を想定したシミュレーションであるということになる．その理由は，1次元配列の要素をランダムに選んで，そこに不具合･欠陥があればその欠陥を修正してデバッグする，というこれらのモデルでは，ある確率で既にテストを行ったプログラム論理について，複数回テストを行うことを許していることとなるからである．言い換えれば，テストの対象となる領域は，本来，テストの進行に対して小さくなることが理想的であるが，これを想定していないモデルとなっている．これに対して，理想的な（質の高い）テスト工程においては，1度テストを行ったプログラム論理に対しては，2度とテストをすることはない，と考えるのが至当であろう.

このことから，ここに述べる3パラメータモデルにおいては，2パラメータモデルでの不完全デバッグの可能性において，3つ目の制御パラメータを導入し，そのパラメータの値が1に近ければ，徐々にテスト領域が縮小するという，質の高い理想的なテスト工程となるようなシミュレーションを，反対に0に近ければ，2パラメータモデルに帰着される，不完全デバッグモデルとなるシミュレーションモデルを構築する．

2.3.1 シミュレーションアルゴリズム(3パラメータ)

我々のモデルにおいて，2パラメータモデルのStep5を以下のStep5”のように変更することで，テスト領域の縮小を考慮したデバッグを想定したモデルを表す.

この新しいパラメータにより，実際のソフトウェア開発工程に近い環境変数を追加することで，更に自由度の高いシミュレーションを行う.

> Step5”: 区間$[1, \mathrm{psize}]$の中からランダムに整数cを選ぶ．もし$\mathrm{p}[c] = 1$ならば$z_i = z_i + 1$とし，$1 - \theta_2$の確率で$\mathrm{p}[c] = 0$に変更する．その後，θ_3の確率でプログラムのブラックボックステスト及びデバッグが行われる．また，$1 - \theta_3$の確率でセルの番号ごとにテスト及びデバッグが行われる．$1 - \theta_3$が1に近づけば近づくほど，全セルに対して順番にテスト及びデバッグが行われる，理想的なデバッグとなる．$1 - \theta_3$が0のとき，完全なブラックボックステストとなる．もし$\mathrm{p}[c] = 0$ならばStep 6に進む$(0 \leq \theta_2 \leq 1), (0 \leq \theta_3 \leq 1)$.

この段階でのθ_2において，前節までと不完全デバッグの意味合いが異なり，θ_2の確率でそのテスト対象となるセル自体のフォールトは修正されるが，新しいフォールトが他のセルに埋め込まれてしまうことを意味する．一方，θ_3が小さければ小さいほど，理想的なデバッグに近づくため，質のよいデバッグとなっていくことを示すことになる．

3. キャリブレーションの方法

実測データを用いた統計的推定を行うため，前提条件として時系列データを出力するプログラムを考える必要がある．また，シミュレーションによるモデルを考えているため，時刻$t_i(i=1,2,3,\ldots,I)$で与えられる時系列データを出力するモンテカルロシミュレーションプログラム（本研究では単にシミュレータと呼ぶ）を考える．まず，$f(t_i|\Theta)$をm個の制御パラメータをもつシミュレータの時刻t_iでの出力結果とする．ここで制御パラメータはm個あるものとし，$\Theta=\{\theta_1,\theta_2,\theta_3,\ldots,\theta_m\}$である．$f$は確率的性質を持っており，シミュレーション結果として我々はベクトル$\mathbf{f}$を得る．すなわち

$$\mathbf{f}=(f(t_1|\Theta),f(t_2|\Theta),f(t_3|\Theta),\ldots,f(t_I|\Theta)), \tag{3.1}$$

である．ここで，$\Theta=\{\theta_1,\theta_2,\theta_3\ldots,\theta_m\}$となる．もし，$\Theta$を固定し$J$回のシミュレーションを行うと，結果としてベクトル

$$\{\mathbf{f}_1,\mathbf{f}_2,\mathbf{f}_3,\ldots,\mathbf{f}_J\}, \tag{3.2}$$

を得ることができる．我々の目的の1つは前節で述べた実際の観測値$(t_i,d_i)(i=1,2,3,\ldots,I)$をうまくシミュレートするようなベクトル$\Theta$の値を逆算することである．

$SE_j(\Theta)$をj番目のシミュレーション結果$\mathbf{f}_j$における誤差二乗和とし

$$SE_j(\Theta)=\sum_{i=1}^{I}(f(t_i|\Theta)-d_i)^2, \tag{3.3}$$

によって与える．ここで$j=1,2,3,\ldots,J$である．ここで，d_iはシステム時刻t_iのときのシステムの実測値である．また，$\mathbf{f}_j$は確率的に振舞うため，$SE_j(\Theta)$の値もjにより変動する．したがって，その標本平均値$MSE(\Theta)$は$SE_j(\Theta)$を用いて次のように与えられる．

$$MSE(\Theta)=\sum_{j=1}^{J}SE_j(\Theta)/J. \tag{3.4}$$

これにより, 制御パラメータのキャリブレーション問題は以下のように与えられる.

$$\tilde{\Theta} = \min_{\Theta} MSE(\Theta). \tag{3.5}$$

ここで$\tilde{\Theta}$は求められた制御パラメータ値からなるベクトルである. $MSE(\Theta)$はΘで微分可能な関数ではないため, $\tilde{\Theta}$を推定するのはあまり容易ではない.

4. 数値例と考察

本節ではいくつかの数値例を用いて，キャリブレーションの様子や，モデルの適合性の比較結果，及び信頼性評価尺度の評価例などを示す．表1，2および図3，4は本節で分析対象とするサンプルデータを表す（それぞれDS-1, DS-2と称す）．Goel and Okumoto[5]に掲載され，そこでは確率過程モデルによって解析されたものである．

表1: ソフトウェアフォールト発見時刻(DS-1).

Fault number k	S_k (days)	Fault number k	S_k (days)
1	9	14	87
2	21	15	91
3	32	16	92
4	36	17	95
5	43	18	98
6	45	19	104
7	50	20	105
8	58	21	116
9	63	22	149
10	70	23	156
11	71	24	247
12	77	25	249
13	78	26	250

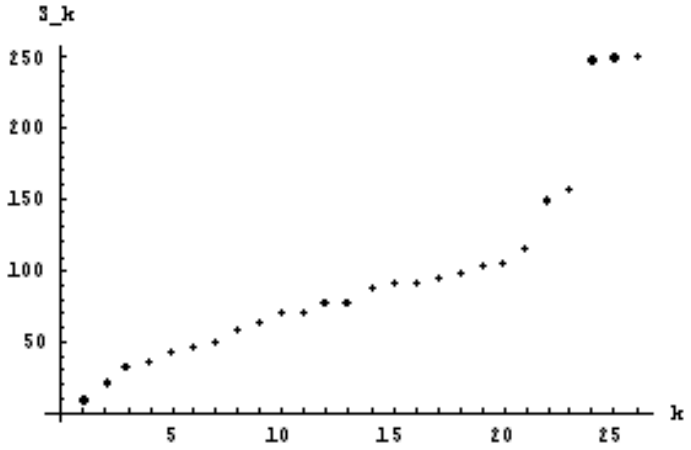

図3: フォールト発見時刻の推移(DS-1).

表 2: ソフトウェアフォールト発見時刻 (DS-2).

Fault number k	S_k (days)	Fault number k	S_k (days)
1	10	15	119
2	35	16	128
3	39	17	128
4	63	18	138
5	66	19	140
6	67	20	149
7	74	21	159
8	77	22	163
9	79	23	191
10	79	24	205
11	80	25	205
12	98	26	220
13	115	27	235
14	117	28	250

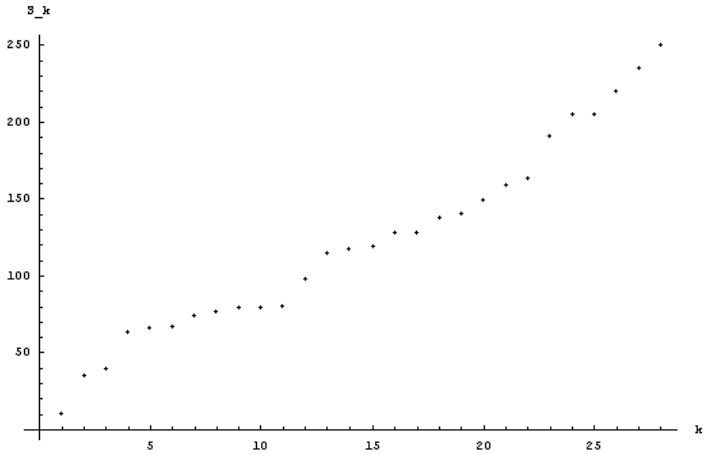

図 4: フォールト発見時刻の推移 (DS-2).

表 2 のDS-2については，例えば故障番号(fault number)$k = 9$及び$k = 10$のときの値は$S_9 = 79$, $S_{10} = 79$となっている．これは 2 つのフォールトが同時に発見･修正されたことを意味しており，従来のNHPPに基づくモデルでは数学的な制約のため，厳密な意味では取り扱うことができないデータであることになる．これに対して，本研究で考察するシミュレーションモデルでは，このようなデータでも取り扱うことができる利点がある．

4.1 DS-1の変換

式(3.3)を計算するために，DS-1を(t_i, d_i)の形式に変換する必要がある．以下のような変形を行う．

Step 1: $S_0 = 0, J = i, K = 1$とする．

Step 2: kが$S_{j-i} \leq k < S_j$を満たすとき，$t_k = k, d_k = j - 1, k = k + 1$とし Step 2を行う．満たさなくなったときStep 3を行う．

Step 3: $j = j + 1$とする．$j < S_k$ならStep 2を行い，そうでなければ Step 4を行う．

Step 4: $t_{S_k} = S_k, d_{S_k} = K$とする．

DS-1を変換したデータを図5に示す．この変換は，生成されたデータが時系列データであるのに対し，元のデータが，n個目のフォールトが発見された時刻を表すものであるため，双方ともに同形式の時系列データでなければ式(3.3)が計算できないために行う必要がある．

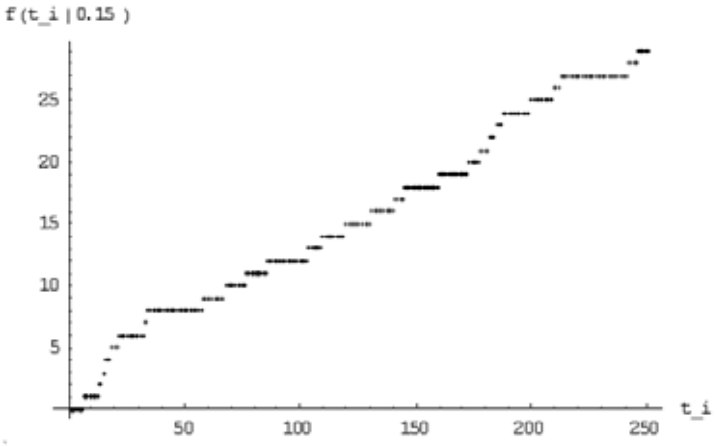

図5: DS-1を変換したデータ．

4.2 実際のシミュレーション結果(DS-1)

前述のシミュレーション方法を踏まえ，パラメータのキャリブレーションを行った．データセットについては前述のDS-1を用い，計算を行った．それぞれ，psize=250, シミュレーション反復回数を$J = 100$とした．

4.2.1 1パラメータモデルの場合

ここまでの準備により制御パラメータを1つもつシミュレータfの制御パラメータのキャリブレーションを行う用意ができた．いま，$J = 100$とするとθ_1を予め与えることにより式(3.4)の$MSE(\theta_1)$の値が得られる．図6にθ_1の値を変えながら$MSE(\theta_1)$を計算した結果を示す.

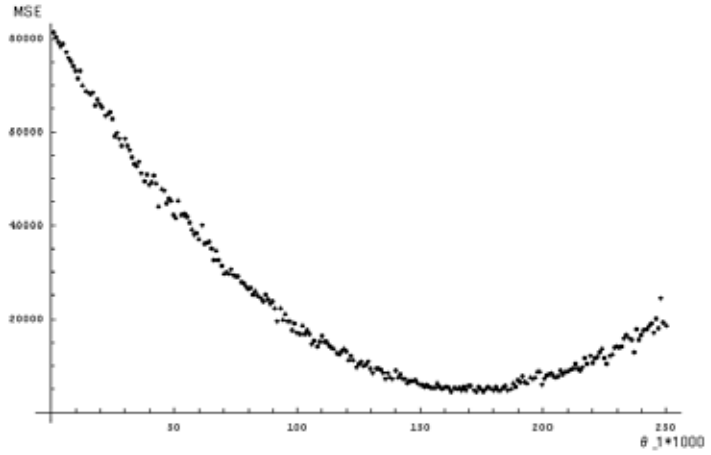

図6: MSEの振舞い(DS-1).

図7より，$MSE(\theta_1)$を(平均的に)最小化するであろうθ_1の値を求めればよいことがわかる．図7の場合は2次曲線を用いて各$MSE(\theta_1)$の値の中心的傾向がほぼ再現されることが見て取れる．したがって，式(3.5)の$\tilde{\Theta}$を得るために2次曲線を当てはめ，それにより最適なθ_1を求めることにした．その結果$\tilde{\theta} = 0.21, MSE(\tilde{\Theta}) = 4409.69$が得られた．この$\tilde{\Theta}$を使って，キャリブレートされたシミュレータ$\tilde{f}$を用いて100回繰り返してシミュレーションを行い，d_iが標準正規分布に従うとの仮定の下で得た平均値の挙動と，95%信頼区間の振舞いを図8に示す.

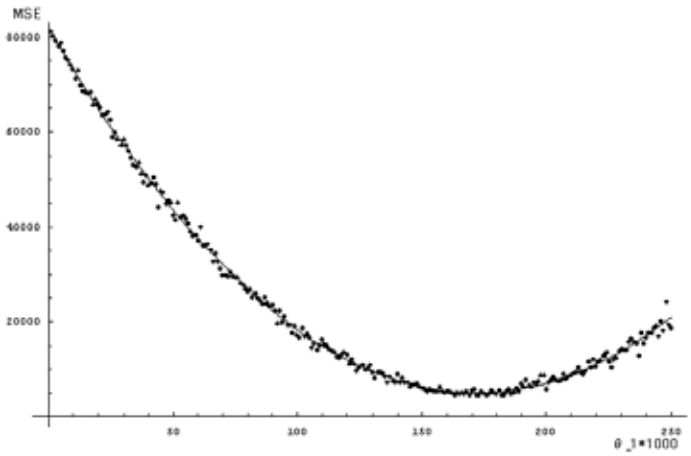

図 7: MSE の 振 舞 い と 2 次 曲 線 の 当 て は め 結 果 (DS-1).

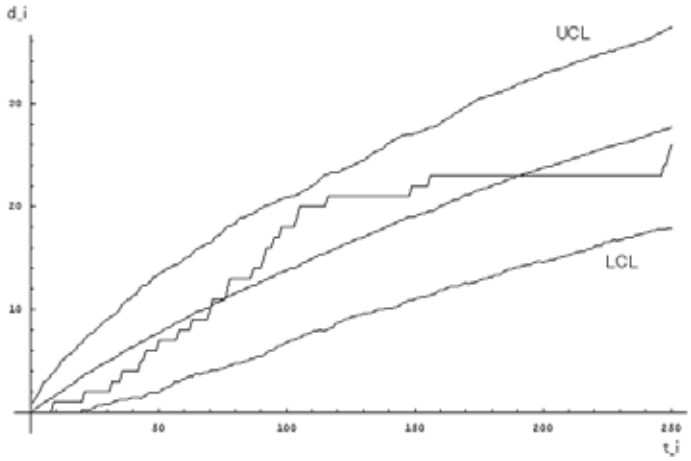

図 8: 1 パ ラ メ ー タ シ ミュレ ー タ の 当 て は め 結 果 (DS-1).

4.2.2 2パラメータモデルの場合

2パラメータモデルに対して，θ_1, θ_2の値を変えながら得た$MSE(\Theta)$を計算した結果を図9に示す. 最適値$\{\tilde{\theta}_1, \tilde{\theta}_2\}$を得るために，ここではプロットした点に曲面を当てはめることなどをせず, 単に$MSE(\Theta)$の最小値を与える$\{\tilde{\theta}_1, \tilde{\theta}_2\}$を探した. これによって，制御パラメータのキャリブレーションの結果として，$\{\tilde{\theta}_1, \tilde{\theta}_2\} = \{0.22, 0.12\}, MSE(\tilde{\Theta}) = 3191.47$を得た. これらの制御パラメータの値を再びシミュレータに与え，図8の場合と同様に95%信頼区間と平均値，および実際のデータの振舞いを図10にプロットした.

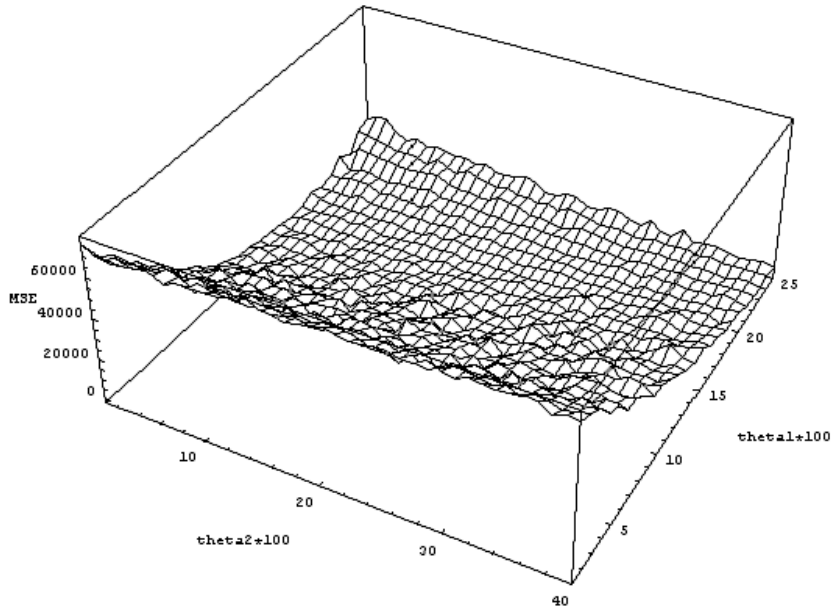

図9: 2パラメータシミュレータによるMSEの振舞い(DS-1).

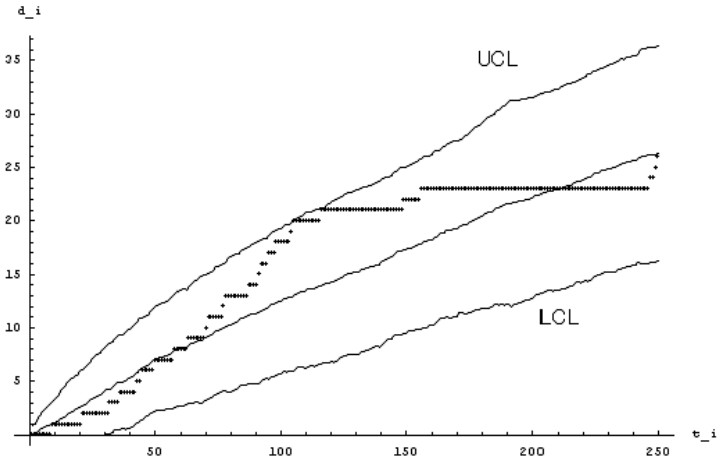

図 10: 2 パラメータシミュレータの当てはめ結果 (DS-1).

4.2.3 3パラメータモデルの場合

ここでは，θ_2, θ_3の値を変えながら$MSE(\Theta)$を計算した結果を示す. 今回は1パラメータ, 2パラメータ時に得た$\tilde{\theta}_1$を踏まえ，$\tilde{\theta}_1$を一定区間，値を変えながら探索することにより, もっとも最適な$MSE(\Theta)$の値を算出した．以下に$\tilde{\theta}_1 = 0.17$として得た結果を図11に示す.

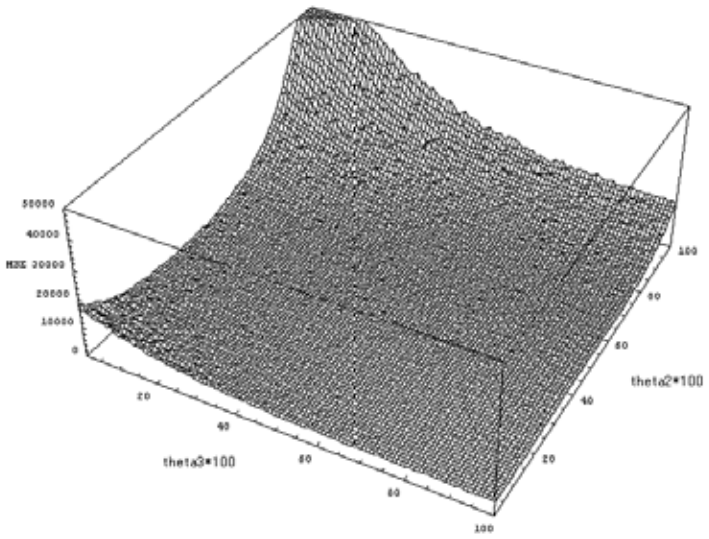

図11: 3パラメータシミュレータによるMSEの振舞い(DS-1, $\tilde{\theta}_3 = 0.17$).

これらにより, 制御パラメータのキャリブレーション結果として，$\{\tilde{\theta}_2, \tilde{\theta}_3\} = \{0.25, 0.3\}, MSE(\tilde{\Theta}) = 3856.78$を得た．

これらの制御パラメータの値を再びシミュレータに与え，図8，10の場合と同様に95%信頼区間と平均値，および実際のデータの振舞いを図12にプロットした．

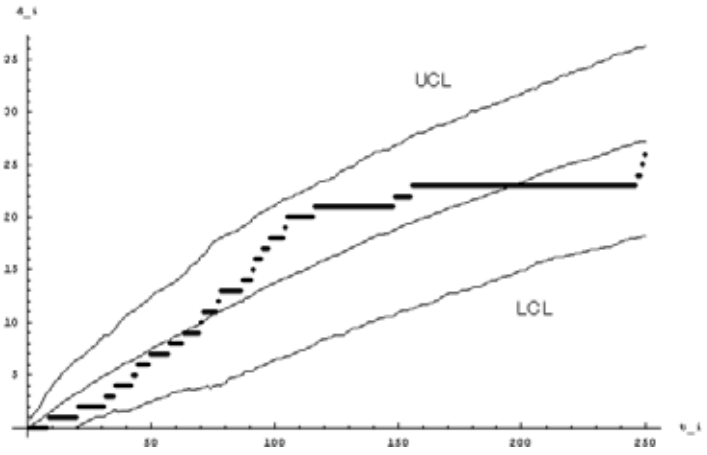

図 12: 3 パラメータシミュレータの当てはめ結果 (DS-1).

4.3 実際のシミュレーション結果(DS-2)

DS-2に対しても前節と同様に数値例を示す. ここでもDS-2を変換した後, パラメータのキャリブレーションを行った. それぞれ同様に, psize=250, シミュレーション反復回数$J = 100$とした.

4.3.1 1パラメータモデルの場合

前節と同様に制御パラメータを1つもつシミュレータfの制御パラメータのキャリブレーションを行う. いま, $J = 100$とするとθ_1を予め与えることにより式(3.4)の$MSE(\theta_1)$の値が得られる. 図13にθ_1の値を変えながら$MSE(\theta_1)$を計算した結果を示す.

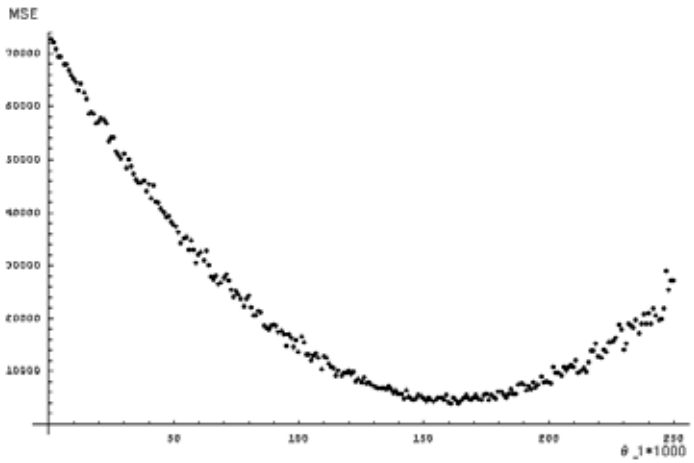

図13: MSEの振舞い(DS-2).

図14より, $MSE(\theta_1)$を(平均的に)最小化するであろうθ_1の値, 式(3.5)の$\tilde{\Theta}$を得るために2次曲線を当てはめ, それにより最適なθ_1を求めることにした. その結果$\tilde{\theta} = 0.16, MSE(\tilde{\Theta}) = 4702.21$が得られた. この$\tilde{\Theta}$を使って, キャリブレートされたシミュレータ$\tilde{f}$を用いて100回繰り返してシミュレーションを行い, d_iが標準正規分布に従うとの仮定の下で得た平均値の挙動と, 95%信頼区間の振舞いを図15に示す.

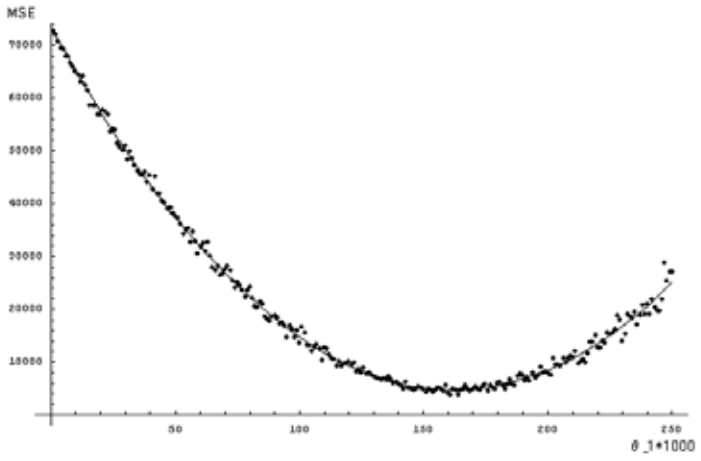

図 14: MSE の振舞いと 2 次曲線の当てはめ結果 (DS-2).

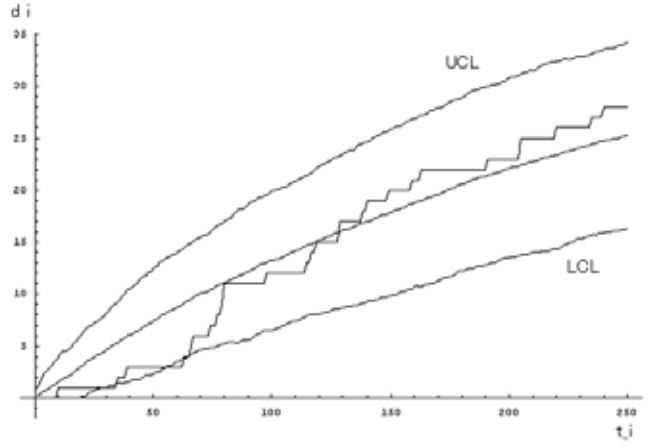

図 15: 1 パラメータシミュレータの当てはめ結果 (DS-2).

4.3.2 2パラメータモデルの場合

今度はθ_1, θ_2の値を変えながら得た$MSE(\Theta)$を計算した結果を図16に示す. 最適値$\{\tilde{\theta}_1, \tilde{\theta}_2\}$を得るために, 前節と同様, プロットした点に曲面を当てはめることなどをせず, 単に$MSE(\Theta)$の最小値を与える$\{\tilde{\theta}_1, \tilde{\theta}_2\}$を探した. これによって, 制御パラメータのキャリブレーションの結果として, $\{\tilde{\theta}_1, \tilde{\theta}_2\} = \{0.22, 0.29\}, MSE(\tilde{\Theta}) = 3010.97$を得た. これらの制御パラメータの値を再びシミュレータに与え, 95%信頼区間と平均値, および実際のデータの振舞いを図17にプロットした.

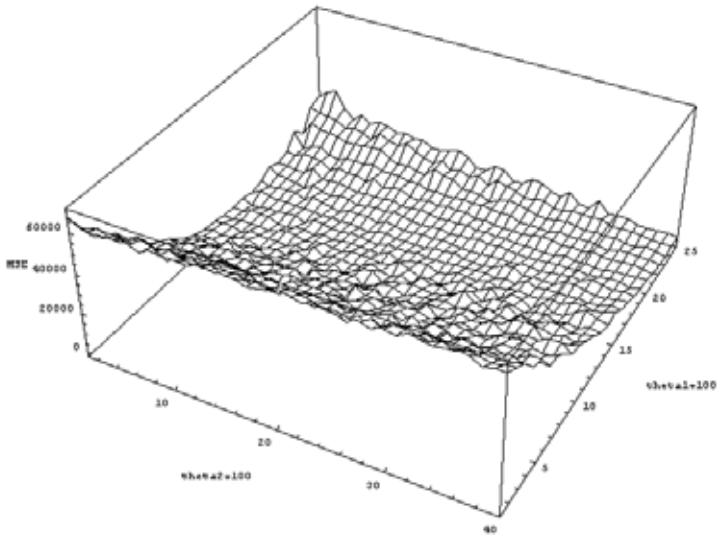

図16: 2パラメータシミュレータによるMSEの振舞い(DS-2).

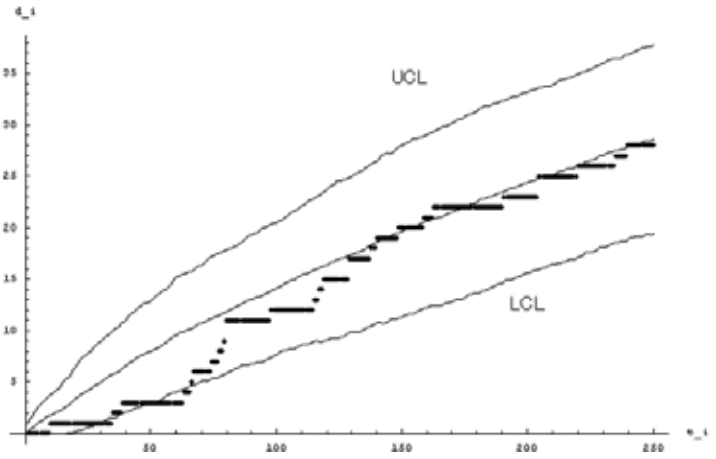

図 17: 2 パラメータシミュレータの当てはめ結果 (DS-2).

4.3.3 3パラメータモデルの場合

θ_2, θ_3の値を変えながら$MSE(\Theta)$を計算した結果を示す. 今回は1パラメータ, 2パラメータ時に得た$\tilde{\theta}_1$を踏まえ, $\tilde{\theta}_1$を一定区間, 値を変え, もっとも最適な$MSE(\Theta)$の値を算出した. 以下に$\tilde{\theta}_1 = 0.17$として得た結果を図18に示す.

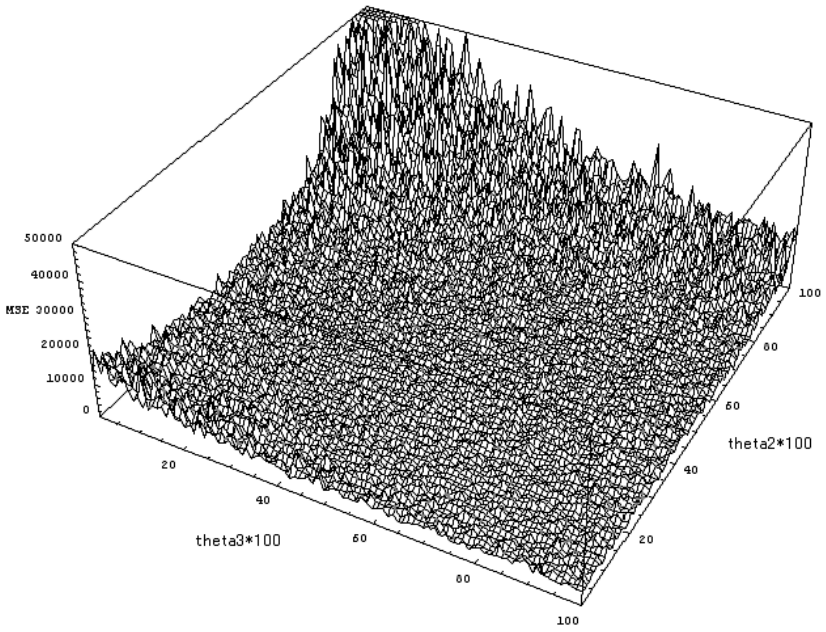

図18: 3パラメータシミュレータによるMSEの振舞い($\tilde{\theta}_3 = 0.17$).

これらにより, 制御パラメータのキャリブレーション結果として, $\{\tilde{\theta}_2, \tilde{\theta}_3\} = \{0.25, 0.35\}, MSE(\tilde{\Theta}) = 2311.97$を得た.

これらの制御パラメータの値を再びシミュレータに与え, 95%信頼区間と平均値, および実際のデータの振舞いを図19にプロットした.

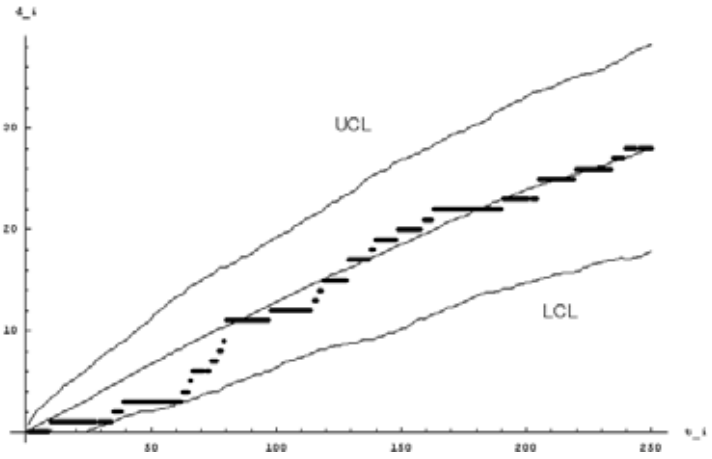

図 19: 3 パラメータシミュレータの当てはめ結果.

4.4 信頼性評価尺度

本節ではソフトウェアの信頼性評価尺度の導出とその評価例を示す. 定量的なソフトウェア信頼性評価尺度として以下の定義によるものを考える.

$$\frac{\text{Number of faults after debugging}}{\text{Number of faults before debugging}} = \frac{\sum_{l=1}^{psize} p[l] \ at \ t_{S_k}}{\sum_{l=1}^{psize} p[l] \ at \ t_{S_k}} \ . \tag{4.1}$$

ここではこの割合を残存フォールト率と呼び, FR_jと表記する. ここで$j = 1, 2, 3, \ldots, J$としJはシミュレーションの反復回数である. FR_jを使って, テスト開始前にソフトウェアプログラム内の潜在フォールト数LF_jを推定することができる. すなわち,

$$LF_j = \frac{K}{1 - FR_j} \ , \tag{4.2}$$

である. ここでKはデータセットにおける実際の総フォールト検出数を表す. さらに, フォールト除去率R_jは以下のように与えることができる.

$$R_j = 1 - FR_j \ . \tag{4.3}$$

4.4.1 モデルの比較

これらのシミュレーションモデルの当てはまりを評価するために, 擬似的なAICを用いる. 本来AICは未知パラメータを最尤法で求めたときに使用可能であるが, 本研究では, 最尤法を用いていないため, 擬似的なAIC(quasi AIC, 以下$qAIC$と呼ぶ)として以下の式を使用する.

$$qAIC = n \times \log \frac{Q_1}{n} + 2 \times (k + 1) \tag{4.4}$$

ここで, nはデータ点数, Q_1は最小の$MSE(\Theta)$, kは未知のパラメータ数を意味する. これを用いて, 1パラメータモデル, 2パラメータモデル, 3パラメータモデルのそれぞれのDS-1時の当てはまりを以下に示す.

$qAIC(\text{one} - \text{parameter simulator}) = 721.52$

$qAIC(\text{two} - \text{parameter simulator}) = 642.69$.

$qAIC(\text{three} - \text{parameter simulator}) = 692.03$

以上より, 2パラメータシミュレータが一番当てはまりが良いことがわかった.

同様にDS-2に対する当てはまりを以下に示す.

$qAIC(\text{one} - \text{parameter simulator}) = 737.58$

$qAIC(\text{two} - \text{parameter simulator}) = 628.14$.

$qAIC(\text{three} - \text{parameter simulator}) = 564.10$

以上の結果より, DS-1では2パラメータモデル, DS-2では3パラメータモデルの当てはまりが良いことがわかった. この結果を受け, 一番当てはまりのよかったモデルのフォールト除去率R_jのヒストグラムを描く.

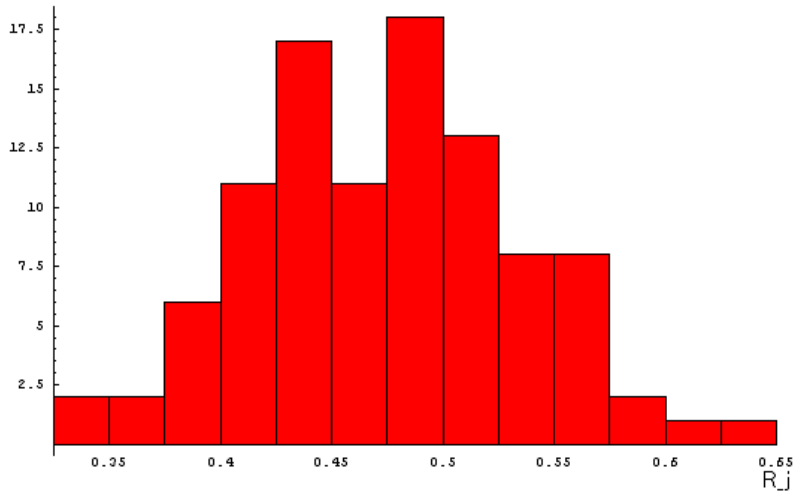

図 20: R_jのヒストグラム(DS-1, 2パラメータ).

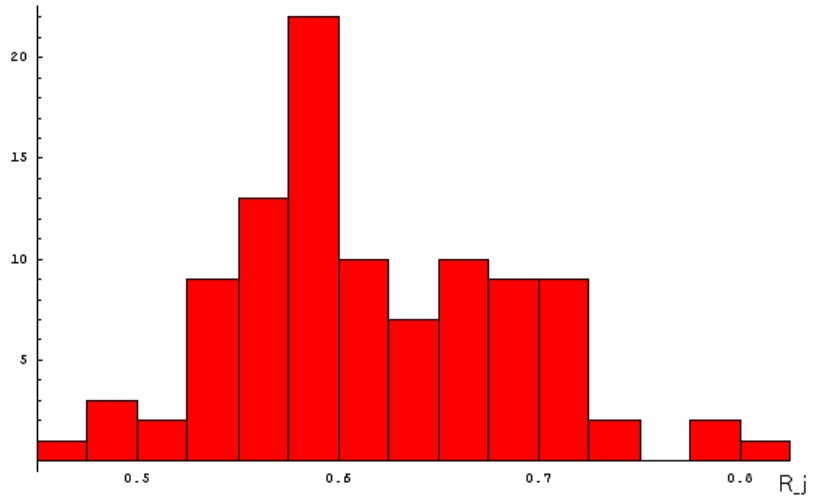

図 21: R_j のヒストグラム (DS-2, 3 パラメータ).

また，それぞれの R_j の期待値 $E[R_j]$ は

$E[R_j](\mathrm{DS}-1) = 47.58$

$E[R_j](\mathrm{DS}-2) = 60.72$,

となり，DS-1, DS-2 の推定フォールト除去率はかなり低いことがわかった．特に，DS-1 を分析した Goel and Okumoto の論文 [5] の結果とは違うものとなっており，シミュレータについて更なる改善の余地があると考えられる．

4.4.2 信頼性評価の結果

前節までに行ったシミュレーション結果及び，評価結果を用いて考察を行う. 本研究でのシミュレーションの利点としては，複数のフォールトが同時に発見･修正される事象に対してそのまま計算できることであることは，前に述べたが，それ以外にも，R_j のヒストグラムを表記できることが挙げられる．ヒストグラムを描くことにより, 従来モデルの場合だと期待値が求められていても，そのバラつきの程度について知ることはあまり容易ではないが，本シミュレーションモデルにおいては，それが容易にわかる．本研究の DS-2 においては，ヒストグ

ラムが0.6付近で突出しているものの，比較的広い値を取っている．つまり，0.5から0.7までほぼ均等に値が取れうるという結果である．これは，フォールト除去率が一定しないものの，5割から7割程度であるということになる．DS-1については，0.4から0.5前後までに値が集中しているため，フォールト除去率が一定するものの，除去率が低いという結果になっている．

さて，シミュレーション結果としてDS-2の場合，3パラメータモデルの$\tilde{\theta}_1 = 0.17$, $\{\tilde{\theta}_2, \tilde{\theta}_3\} = \{0.25, 0.35\}$という値が最適値となった．これは，フォールトが17%プログラム中に含まれ，4回に1回不完全デバッグが起こり，テスト領域に関しては35%ほど2度とテストを行わない質の高いテストを行うという状況である．質のパラメータについて，$\tilde{\theta}_3 = 0.35$という結果が出ているので，それに応じたソフトウェア開発におけるコストの配分をすることも可能であり，このようにして本研究のモデルが実際のソフトウェアテストの管理における情報提供の一助となると考えられる．

5. 終わりに

本研究では最小二乗法に基づいたモンテカルロシミュレーションモデルに含まれる制御パラメータのキャリブレーションの方法について述べた．具体的には，この方法をソフトウェア信頼性評価において用いることのできる単純なシミュレーションモデルについて適用し考察した．本研究で提案した方法は，発見的な方法によって（平均的に）最適な制御パラメータの値をもつシミュレータを見つけることができるが，制御パラメータの個数がさらに大きくなったときに，効率よくを算出する方法についてさらに研究しなければならない．また，我々はこれらのシミュレーションモデルのノイズが結果に及ぼす影響についても調べる必要がある．これは4節においてpsize=250と与えたパラメータの値については，任意にそれを定めており，パラメータpsizeの最適値について何ら考慮していない．しかし実際にはある評価尺度の下で最適なpsizeの値があると考えられる．このことから，前節で求めたフォールト除去率の推定結果がかなり低い値を示したが，これは不適当なpsizeの値が影響したと考えることもでき，これらの点については改善の余地がある．

謝　辞

本研究を進めるにあたり, 本学工学部経営工学科木村光宏教授から全面的なご指導ご協力を賜りました. また, 副査を引き受けて下さった本学工学部経営工学科若山邦紘教授に併せて感謝致します. 信頼性工学研究室の諸先輩方, 同輩からも様々な助言, 助力を頂きました. ここに深くお礼申し上げます.

参考文献

[1] H. Pham, *Software reriability*, Springer-Verlag, Singapore, 2000.

[2] M. Lyu, *Handbook of software reliability engineering*, IEEE Computer Society Press, Los Alamitos, 1995.

[3] J. D. Musa, *Software Reliability Enginnering*, McGraw-Hill, New York, 1998.

[4] M. Sahinoglu, "Compound Poisson software reliability model, "IEEE Trans. Software Engineering, vol.18, pp. 624-630, 1992.

[5] A. L. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measure, "IEEE Trans. Reliability, R-28, pp. 206-211, 1976.

[6] 中嶋浩人, 木村光宏,「ソフトウェアデバッグシミュレーションモデルにおける未知パラメータのキャリブレーション法に関する一考察」, 電子情報通信学会技術研究報告, R2007-17, pp. 19-24, 東京, 2007.