Article ID: 1007-1202(2000)01-0035-06

Singularity of Some Software Reliability Models and Parameter Estimation Method

XU Ren-zuo, ZHOU Rui, YANG Xiao-qing

State Key Laboratory of Software Engineering, Wuhan University, Wuhan 430072, China

Abstract: **According to the principle, "The failure data is the basis of software reliability analysis", we built a software reliability expert system (SRES) by adopting the artificial intelligence technology. By reasoning out the conclusion from the fitting results of failure data of a software project, the SRES can recommend users "the most suitable model" as a software reliability measurement model. We believe that the SRES can overcome the inconsistency in applications of software reliability models well. We report investigation results of singularity and parameter estimation methods of experimental models in SRES.**

Key words: **software reliability measurement models; software reliability expert system; singularity; parameter estimation method; path following method; maximum likelihood ML-fitting algorithm**

CLC number: **O 213.2**

After the software failure data collected in software testing being statistically processed by the software reliability models, the reliability targets like the current reliability and MTTF can be calculated and estimated. Moreover, software reliability models can help software development managers to set down the targets like the termination time of software testing and the best time for the software production to be released. Software reliability models play a chief role in the management of the cost and the rate of progress of a software project.

In the process of using software reliability models, the first problem that users meet is how to select model correctly, and the second one is how to estimate the values of the model's parameters. The influence of the two problems is often mentioned in external document[1], which will be mainly discussed following.

1 Inconsistency of Software Reliability Models

It has been more than twenty years since the establishment and the application of software reliability fixed quantification estimation models were studied. Today, one of the achievements in the field is that a batch of software reliability models[2~6], which can be used in practical projects, have been built.

Probability pattern of software failure behavior of a specific model, and also limit various ways of descriptions of every model to the software failure behavior, and various testing and rectifying methods used by software developers and checkers. These hypotheses are the rationale of every software reliability model. They determine not only every model's accommodation but also the scope and the depth of every model's application.

The problem of inconsistency in evaluation of software reliability mainly contains two sides. First, because every software reliability model (SRM) has specific hypothesis (for example, the description of random behavior of software failure), and the random distribution that are often used in mathematical statistics have all been used by these available models, some results would have been obtained by using correct

Received date: 1999-06-10
Foundation item: Supported by the National Natural Science Foundation of China
Biography: XU Ren-zuo(1946-), male, Professor, research interests include software engineering, software reliability engineering, software quality guarantee technology, reliability theory, software safety and security.

statistical method in dealing with the data collected during software development (in particular, software testing), therefore, choosing different models will lead to different results, which results in the problem of inconsistency of software reliability evaluation. Second, in software testing, with the progress of software testing, the number of the collected failure data will also increased continuously. At the time of T_1, the model M_1 may be selected to evaluate software reliability, but at the time of $T_2(T_2 > T_1, \triangle t = T_2 - T_1)$, another model M_2 or M_1 may be selected to evaluate software reliability. Thus, the problem of inconsistency in another sense is produced.

In general, we use $\lambda(t)$ to indicate the risk function. $\lambda(t)$ is the probability of program failure occurring per unit time when the program correctly runs to the time of t (in fact, $\lambda(t)$ should be probability density, the real probability should be $\lambda(t)\triangle t)$.

If T indicates the time from 0, from which a program begins to run, to the time of program failure occurring, the values of T are apparently different for different executions. So it can be concluded that T is a continuous random variable. With the result that the formula is:

$$\lambda(t)\Delta t = Pr\{t < T \leqslant t + \Delta t | T > t\}$$

The formula above describes the relationship between the risk function $\lambda(t)$ and the reliability function $R(t)$. The risk function $\lambda(t)$ is paid widespread attention to in the study of software reliability. Because from the formula above we can know that $R(t)$ can be calculated directly as soon as the expression of $\lambda(t)$ is available. But in practical work, the relationship between $\lambda(t)$ and t is very complex, so people can not get the exact expression of $\lambda(t)$ about t. In current software reliability models, people made kinds of hypotheses about $\lambda(t)$, which leads to the inconsistency of the application of software reliability models theoretically.

To overcome the harm of inconsistency, we must overcome the willful selection of the SRM (Software Reliability Model). That is to say every user tries to select "the most suitable model". The SRES (Software Reliability Expert System-SRES)[7,8] developed by us just helps users to correctly select SRM to evaluate the software system's reliability by correctly offering software failure data of software system according to system's indication under the circumstance that users don't completely understand the SRM. The offered software failure data must really and exactly reflect the history and behavior of failure of software system. Because the SRES developed by us follows the unchangeable principle: "Let failure data explain everything." All the inference and judgment that SRES makes must be based on the pre-analysis of software failure data offered by users. We choose CLIPS[9] as the development circumstance of SRES.

As far as the rationale of our SRES is concerned, it is an optimization problem of the pre-analysis of software failure data offered by users under the restraint of a group of rules of discrimination between good and bad. The optimization process is optimizing the results of fitting the software failure data offered by users by means of the specific experimental model, and in the light of the optimized results, "a most suitable model", which can estimate the failure behavior of software system in a specific time of future, can be selected.

2 Experimental Models Library

When developing SRES, we should build an experimental model library that can be enlarged and has many typical experimental models. After long-term study and comparison, finally we selected following models as experimental models[6,7,10~13]: Weibull Model-WM (incomplete data), Geometric Model-GM (special data), Modified GM-MGM (special data), Geometric Poisson Model-GPM (special data), Hypergeometric Distribution Model-HGDM (special data), Goel & Okumoto Model-GOM (complete data, incomplete data), Yamada & Osaki Model-YOM (complete data, incomplete data), Goel Three Parameters Model-Goel(3) (complete data, incomplete data), Musa & Okumoto Logarithmic Poisson Executing Time Model-MOM (complete or incomplete data of executing time), Parameter Adjusting and Three Parameters NHPP Model-NHPP3AD (incomplete data), Littlewood's Bayesian Debugging Model-LDM (complete data), Littlewood & Verrall's Linear and Quadratic Growth Models-LVLM (complete data), and LVQM (complete data), Schneidewind Models-SM (special data), Modified SM-MSM(complete data). These fifteen models make up the experimental model library of SRES. Here, complete data mean the failure interval data, incomplete data mean cumulative failure data. The reason why we make a distinction between complete and incomplete data is that SRES should have the ability to identify failure data types automatically. When realizing some models in experimental models library, we encountered a type of problem named singularity.

3 Singularity Problem of Certain Experimental Models

The system of equations of parameter estimation of Software Reliability Model(SRM):

$$g_i(t,\theta)=0,\ (i=1,\cdots,m) \tag{1}$$

Where, m is the number of the model's parameters. Due to the incomplete consideration of the problem or the limitation of some conditions during the modeling software reliability for a software project, in Eq. (1) we probably introduce singularity[14,15]. Now we definite the singularity as following:

Definition If there is a group of numbers that are not zero totally, and, the following equation is tenable:

$$\sum_{i=1}^{m} k_i g_i(t,\theta)\equiv 0 \tag{2}$$

We say that the SRM is singular (m is the number of the model's parameters).

The singularity of the model originated from the fundamental limit during the course of building the model comes from the incompleteness of the certain formulations of the model. Because of existing singularity there are innumerable parameters θ which satisfy the system of Eq. (1). That is to say we can use innumerable curves to fit the failure data. Then which fitted curve should we select to evaluate and predict software reliability? So we must have effective method to deal with this kind of singularity models.

In our SRES, we have met three singularity models. They are NHPP Three Parameters Model of Non-homogeneous Poisson Process (NHPP3AD), Bayesian LV Models, including linear model LVLM and quadratic model LVQM.

3.1 The Singularity of NHPP3AD

In the estimation equations of NHPP3AD using incomplete data to find out the model's parameters:

$$\begin{cases}(1-B)\hat{a}-y_m\cdot C=0\\ t_m\cdot\hat{a}\cdot B+AC=0\\ (t_m\hat{b}\cdot BC+B-1)\hat{a}+y_m\cdot C+\hat{b}\,AC^2=0\end{cases} \tag{3}$$

Where: $A=\sum_{i=1}^{m}\left[(y_i-y_{i-1})\dfrac{t_{i-1}e^{-\hat{b}t_{i-1}C}-t_ie^{-\hat{b}t_iC}}{e^{-\hat{b}t_{i-1}C}-e^{-\hat{b}t_iC}}\right]$, $B=e^{-\hat{b}t_mC}$, $C=1-\hat{\beta}$

Looking into the three equations in the system of equations above we can find: the third equation is the combination sum of the first two equations, which satisfies the definition of singularity. So in fact, only the first two equations in this system of equations are valid. Therefore, the system of equations is equivalent to the system of equations as following:

$$\begin{cases}(1-B)\hat{a}-y_mC=0\\ t_m\hat{a}\,B+AC=0\end{cases} \tag{4}$$

Now, there are three parameters but only two equations. So its solution is indefinite[14].

3.2 The Path Following Method

To find its solutions, we use path following method[16]. Its first two equations can be written as:

$$\begin{cases} f_1(a,b,\beta) = 0 \\ f_2(a,b,\beta) = 0 \end{cases} \tag{5}$$

They are equivalent in the following meaning: if (a, b, β) is the solution of Eq. (3), it is the solution of Eq. (5), and vice-versa.

Because it has just one degree of freedom, generally, the solution of Eq. (5) is some curves. Assume its one solution curve is $\{a(s), b(s), \beta(s)\}$, $s \in [0, S)$, which is a arc length parameter and $a(0) = a_0$, $b(0) = b_0$, $\beta(0) = \beta_0$ is a known solution, then the solution curve's tangent direction $(\dot{a}_0, \dot{b}_0, \dot{\beta}_0)$ at point (a_0, b_0, β_0) can be determined by constructing the following system of equations:

$$\begin{cases} \frac{\partial f_1}{\partial a}\dot{a}_0 + \frac{\partial f_1}{\partial b}\dot{b}_0 + \frac{\partial f_1}{\partial \beta}\dot{\beta}_0 = 0 \\ \frac{\partial f_2}{\partial a}\dot{a}_0 + \frac{\partial f_2}{\partial b}\dot{b}_0 + \frac{\partial f_2}{\partial \beta}\dot{\beta}_0 = 0 \\ \dot{a}_0^2 + \dot{b}_0^2 + \dot{\beta}_0^2 = 1 \end{cases} \tag{6}$$

After $\dot{a}_0$, $\dot{b}_0$, $\dot{\beta}_0$ being obtained, select a proper step length $\triangle s$ and predict:

$$\begin{cases} a_1^0 = a_0 + \triangle s \cdot \dot{a}_0, \\ b_1^0 = b_0 + \triangle s \cdot \dot{b}_0, \\ \beta_1^0 = \beta_0 + \triangle s \cdot \dot{\beta}_0 \end{cases}$$

We can get the initial approximation of the next point's coordinates on the solution curve, and then use Newton's method to find the solutions of extension system

$$\begin{cases} f_1(a_1,b_1,\beta_1) = 0 \\ f_2(a_1,b_1,\beta_1) = 0 \\ (a_1 - a_0)\dot{a}_0 + (b_1 - b_0)\dot{b}_0 + (\beta_1 - \beta_0)\dot{\beta}_0 = \triangle s \end{cases} \tag{7}$$

Taking its solution a_1, b_1, β_1 as the next point's coordinates on the solution curve and (a_1, b_1, β_1) as the new starting point, we can work out all the points on the solution curve by doing so extensively.

In the process of deriving algorithm above, we assumed that we knew an initial solution (a_0, b_0, β_0), but because extension system of Eq. (7) is well posed, the assumption is unnecessary in fact. For most arbitrary point (a_0, b_0, β_0), Newton's iterative process to find the solutions of equation (7) all can well converge on one solution (a_0, b_0, β_0) of equation (5) which is regarded as genuine starting point for extending.

In practical problems, the accumulative failure data we used are listed in Table 1. They are collected in testing phrase when we developed the software WPADT.

Table 1 Accumulative Failure Data: D. inc

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$t(i)$	0	2	3	7	8	9	10	11	18	21	33	35	37	44	45
$y(i)$	0	4	5	7	8	14	17	28	29	30	31	33	41	46	48
i	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29
$t(i)$	47	48	49	50	51	52	53	55	56	57	63	76	83	91	106
$y(i)$	50	53	56	59	64	67	68	69	71	74	76	78	79	80	81

Taking the data from $i = 0$ to $i = 26$, using path following method, we can get a group of parameter estimation values as following. They are listed in Table 2, where,

$$F_{\max} = \max(|f_1(a,b,\beta)|, |f_2(a,b,\beta)|)$$

We can find out (a, b, β) in Table 2 are all the solutions of Eq. (5).

Table 2 Parameter Estimation Values

$\hat{a}$	$\hat{b}$	$\hat{\beta}$	$F_{\max}$
70.7973193841682900	0.0223293392111892	0.4481236104565089	0.7815970093361*D*-13
84.0469163682510800	0.0188092249910214	0.3448408900076743	0.7518430322762*D*-12
87.0468251562265000	0.0181609996336428	0.3214561228262518	0.9177103521552*D*-12
93.0466427579045800	0.0169899451814203	0.2746865882623570	0.7833733661755*D*-12
102.0463692072263000	0.0154915591025266	0.2045322860529145	0.8579803534303*D*-12
108.0461868619497000	0.0146313109762203	0.1577627510760219	0.9070522111188*D*-12
120.0458222059607000	0.0131687823091156	0.0642236808528039	0.6661338147751*D*-14
123.0457310472698000	0.0128477221136968	0.0408389132556379	0.9263700917472*D*-12

The problem's solution curve is shown in Fig. 1. In practical working process we used the character: $a(y(n))$, to reduced the scope of extending, and omitted much unnecessary calculating.

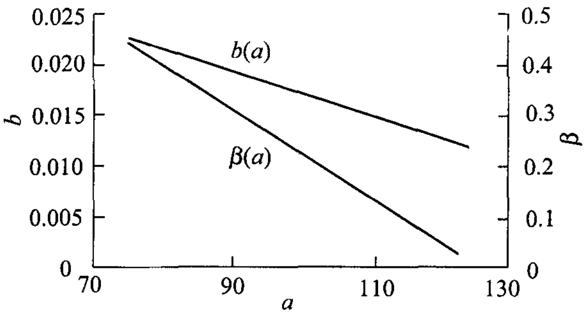

Fig. 1 Solution Curve of the Problem
$b(a)$ is the solution curve of b to a,
$\beta(a)$ is the solution of β to a

It can be found out from the results above that by means of path following method we not only successively solved the singularity problem of maximum likelihood system of equations, that is to say, replaced the former ill posed problem with a well posed problem, thus by fewer several iteration it can reach exact solution which can't be reached before, but also get all probably solutions of parameter estimation equation. The shortcoming of path following method is that there are numerous calculations in the parameter estimation process. For overcoming it, we introduce the so-called ML-fitting algorithm.

3.3 ML-fitting Algorithm

Because of no in A, B, C, and there are existing the relationship of: $0<B<1$, $C\neq 0$, we can make equivalent transformations, and get:

$$\begin{cases} t_m y_m B + (1-B)A = 0 & (8) \\ \hat{a} - \dfrac{y_m C}{1-B} = 0 & (9) \end{cases}$$

Notice that A, B are all the functions of $\hat{b}$, $\hat{\beta}$, if we let: $x = \hat{b}\,C = (1-\hat{\beta})\hat{b}$, then the system of Eq. (3) can be converted to formal one-variable equation to x:

$$f_1(x) = t_m y_m e^{-xt_m} + (1-e^{-xt_m}) \cdot \sum_{i=1}^{m} (y_i - y_{i-1}) \frac{t_{i-1}e^{-xt_{i-1}} - t_i e^{-xt_i}}{e^{-xt_{i-1}} - e^{-xt_i}} \tag{10}$$

According to the actual background it can be known that $x \in (0, 1)$. So we can firstly find the root x of Eq. (10), then for arbitrary $\hat{\beta} \in (0,1)$, find out: $\hat{b} = x/(1-\hat{\beta})$, then according to: $\hat{a} = y_m(1-\hat{\beta})/[1 - e^{-xt_m}]$, we can find out $\hat{a}$. Finally we regard the defined fitting rate as the constraint of the optimizing process, let: $G_{\hat{a},\hat{b},\hat{\beta}} = [1 - e^{-xt_m}]$, to make the following optimizing process and find out:

$$G_{\bar{a},\bar{b},\bar{\beta}} = \min_{(\hat{a},\hat{b},\hat{\beta})} \{G(\hat{a},\hat{b},\hat{\beta})\} \tag{11}$$

to satisfy $\bar{\theta} = (\bar{a},\bar{b},\bar{\beta})$ of Eq. (11), and regard it as the parameter value to find. We define the algorithm above as maximum likelihood-fitting algorithm[15].

For the method of how to find out the solution x discussed above, we can adopt method of interval

0. 618. That is to divide the interval [0,1] into p length-equal subintervals:

$$(\tilde{a}_{i-1}, \tilde{a}_i),\ i = 1,2,\cdots,p, a_0 = 0, a_p = 1$$

When p is large enough, in every subinterval there is only one root at most that makes $f_1(x) = 0$.

Assume: there is one root in the subinterval $[g,\ h]$, that has roots. Method of interval 0. 618 is to take $c = g + 0.382(h - g)$, $d = g + 0.618(h - g)$ as the division points of $[g,\ h]$.

If $|f(c)| < |(f(d)|$, the root is in $[g_1, h_1] = [g, d]$, the former point c in $[g_1, h_1]$ is the right division point d_1 in the new subinterval;

If $(|f(c)| > |(f(d)|$, the root is in $[g_1, h_1] = [c, h]$, the former point d in $[g_1, h_1]$ is the left division point C_1 in the new subinterval;

Repeat, until $|f(c_i)| < \varepsilon$ or $|f(d_i)| < \varepsilon$ or $h_i - g_i < \varepsilon$

Select p is large enough, we can avoid the condition which make the method of interval 0. 618 ineffective. The parameter values found by maximum likelihood-fitting algorithm:

$$\hat{a} = 58.59744,\quad \hat{b} = 0.03342608,\quad \hat{\beta} = 0.348500.$$

References:

[1] Poore J H, Mills H D, Mutchler D. Planning and Certifying Software System Reliability. *IEEE Software*, January, 1993: 88-99.

[2] XU Ren-zuo, XIE Min, ZHENG Ren-jie. *Software Reliability Models and Applications*. Beijing: Tsinghua University Press, Naning: Guangxi Science&Technology Press (In Chinese), 1994.

[3] Meller P. Software Reliabilty Modelling: the State of the Aat. *Information and Software Technology*, 1987, **29**(2):26-35.

[4] MUSA J D, IANNINO A, OKUMOTO K. *Software Reliabilty: Measurement, Prediction, Application*. New York: McGraw-Hill, 1987.

[5] FARR W H. A Survey of Software Rellabilty Modeling and Estimation. NSWC-TR-82-171, NAVAL SURFACE WEAPONS CENTER, September 1983.

[6] American National Standard: ANSI/AIAA R-013-1992, Recommended Practice for Software Reliability, American Institute of Aeronautics and Astronautics, and American National Standards Institute, ratified version, Feb 23, 1993.

[7] XU Ren-zuo. *Software Reliability Expert System and Realization, Proceedings of the Fifth Simposium on Reliability*. Beijing: Mechanical Industry Press, 1995, 227-231(Ch).

[8] XU Ren-zuo. *Development of Software Reliability Expert System (SRES)*. Beijing: Tsinghua University Press, 1996 (Ch).

[9] Gonzalez A J, Dankel D D. The Engineering of Knowledge-Based Systems-Theory and Practice. New Jersey: Prentice-Hall International Inc, 1993.

[10] Goel A L, Okumoto K. Time Dependent Error Detection Rate Model for Software Reliability and Other Performance Measures. *IEEE Transactions on Reliability*, 1979, **28**(3):206-211.

[11] Yamada S, Obha M, Osaki S. s-Shaped Software Reliability Growth Models and Their Applications. *IEEE Transactions on Reliability*, 1984, **33**(4):289-292.

[12] Ohba M, CHOU Xiao-mei. Does imperfect debugging affect software reliability growth? *Proceedings of 11-th ICSE*, Los Alamitos, USA:IEEE Computer Society Press, 1989. 237-244.

[13] Littlewood B, Verrall J. A Bayesian Reliability Growth Model for Computer Software. *The Journal of the Royal Statistical Society*. Series C, 1973, **22**(3):45-50.

[14] LIU Chun-lei, XU Ren-zuo. The Singularity Problem in Non-homogeneous Poisson Process Models Parameters Estimations. *Journal Wuhan University (Natural Science Ed)*. 1993, (5):21-26 (Ch).

[15] HU Zhe-min, XU Ren-zuo. Parameter Computational Methods in Nonhomogeneous Poisson Process Models. *Journal Wuhan University (Natural Science Ed)*, 1995, **41**(3):321-328 (Ch).

[16] Kubicek M, Marek M. *Computational Methods in Bifurcation Theory and Dissipative Structure*. New York: Springer-Verlag, 1983.