A Fast Algorithm to Compute Maximum Likelihood Estimates for the Hypergeometric Software Reliability Model

Frank Padberg
Fakultät für Informatik
Universität Karlsruhe, Germany
padberg@ira.uka.de

Abstract

We present a fast and exact algorithm to compute maximum likelihood estimates for the number of faults initially contained in a software, using the hypergeometric software reliability model. The algorithm is based on a rigorous mathematical analysis of the growth behavior of the likelihood function for the model. We also clarify the stochastic process underlying the model and prove a recursion formula which is central for most previous work on the hypergeometric software reliability model.

1 Introduction

The number of faults contained in a software is an important quality attribute of the software, not only in theory, but also in practice. For example, in order to decide when to stop testing and debugging a software, project management must know how many faults still remain in the software after each testing and debugging cycle. How many faults were fixed during debugging is known, whence management might as well ask for the number of faults *initially* contained in the software. Clearly, it is impossible to determine the number of faults contained in a software exactly. Therefore, it must be *estimated* in practice.

The idea behind probabilistic software reliability modeling is to regard the software tests performed during development as a large *random experiment.* The random experiment is formally modelled as a stochastic process which is parameterized in some way by the number m of faults initally contained in the software. The particular results observed in the software tests then are used to estimate the value of m by applying suitable statistical techniques.

A well-known example for a probabilistic software reliability model is the *hypergeometric reliability model.* This model has been studied in a number of papers [5-14, 18-21]. In the hypergeometric model, the result of software test k consists of the number w_k of faults detected in the test and the number x_k of those faults which are newly detected. A fault is newly detected if it has not already been detected in one of the earlier tests. Each test is modelled using a suitable hypergeometric probability distribution, which explains the name of the reliability model; see subsection 2.1. The hypergeometric distributions all depend on the number m of faults initially contained in the software. The distributions for the individual tests are combined into a stochastic process which describes the whole series of tests. The stochastic process for the hypergeometric reliability model has not been properly formalized in previous papers, although this is necessary in order to correctly apply statistical estimation techniques. Therefore, as a first step we clarify the stochastic process underlying the model in subsection 2.2.

A standard estimate in statistics which frequently is computed in software reliability models other than the hypergeometric model (such as [4, 15, 22]) is the *maximum likelihood estimate.* To compute the maximum likelihood estimate in some reliability model, one must find a global maximum of the likelihood function $L(m)$ for the model (see subsection 3.1 for the definition) and particular test series result. Usually, one attempts to solve the maximum likelihood equation

$$\frac{\mathrm{d} \log L(m)}{\mathrm{d}\, m} = 0.$$

In the hypergeometric reliability model, the formula for the likelihood function $L(m)$ contains a large number of binomials. Thus, the maximum likelihood equation is complicated. For the special case that each test reveals the same number of faults (w_k is constant), the approach of solving the max likelihood equation has been followed for the hypergeometric model in [21]. Several methods are used in [21] to approximate the derivative of the logarithm of the likelihood function, for example, approximating the factorials using Stirling's formula. For each method used in [21], the resulting approximate maximum likelihood equation is too complicated to be solved exactly; Newton's method is used. Approximating the equation and its zeros would require to give an error bound for the solution if it were to be used as an approximate maximum likelihood estimate, but no error bounds are given. Also, the approach not always terminates with a solution. The question whether a maximum likelihood estimate exists in the hypergeometric reliability model for each given test series result is not addressed in [21].

In this paper, we show that for each but one special kind of test series maximum likelihood estimates exist in the hypergeometric software reliability model. We present a *fast and exact algorithm* for computing the maximum likelihood estimates. The key idea leading to our algorithm is to study the *growth quotient*

$$Q(m) = \frac{L(m)}{L(m-1)}$$

of the likelihood function $L(m)$ instead of the likelihood function itself. Clearly, the likelihood function $L(m)$ is increasing if and only if the growth quotient $Q(m)$ is greater than one; thus, no information is lost. The growth quotient for a single hypergeometric distribution has already been studied long ago, see [3]. Studying the growth quotient has the advantage that the binomials in the formula for the likelihood function, which make it difficult to study the growth behavior of the likelihood function directly, cancel almost completely. The growth quotient turns out to be a rational function, see subsection 3.2, and we study its growth behavior by means of basic algebra and advanced calculus. A typical graph of the growth quotient is shown in figure 1: the graph crosses the line $y = 1$ once, coming from above, and runs below the line after having crossed it. To compute the maximum likelihood estimate one must determine the crossing point. The resulting algorithm is presented in section 4.

Instead of maximum likelihood estimates, a least squares method has been used with the hypergeometric reliability model to estimate the number of faults initially contained in the software, see [13, 19]. The least squares method is based on a *recursion formula* for the expected value of the cumulative number c_k of different faults detected during the first k tests. The recursion formula has not been derived properly in previous papers, see subsection 2.3. We prove rigorously in the paper that the recursion formula holds without any restriction. The recursion formula is important since most previous work on the hypergeometric reliability model is based on it.

Some difficulty arises when trying to compute the maximum likelihood estimates for the examples given in previous papers. In these examples, the number w_k of faults detected in test k has never been recorded. Since the w_k are essential parameters in the hypergeometric reliability model, the model can't be directly applied to the examples. Previous papers assume that the w_k can be computed using a function which takes the number k of the test as input. The function is interpreted as a "learning curve", see [6]. The learning curve often takes some additional input, such as the number u_k of testers involved in test k. The learning curve often is parameterized by the number m of faults initially contained in the software, and some other parameters. The goal in previous papers then has been to estimate both m and the other parameters of the learning curve from the test observations x_k (and the input u_k) using the least squares method. Several functions have been proposed as learning curves, see [6, 14].

From our point of view, the learning curve approach intertwines two problems which should better be treated separately. The first problem is to estimate the number of faults initially contained in the software from the empirical test results, which include both the numbers x_k and w_k when using the hypergeometric reliability model. It does not make much sense to use only part of the test results for estimating; writing down the w_k is at no cost while running the tests. The second problem is to study how the number of faults detected in a test relates to the number of faults contained in the software, including any learning effects during software testing. Dependencies between different faults and the kind of test cases used should be considered, too. The second problem should be addressed only *after* an estimate for the number of faults in the software has been computed. Consequently, we do not assume that the w_k can be computed using some learning curve, but rather take them as input for our algorithm.

In section 5, we use our algorithm to compute the maximum likelihood estimates for two examples taken from previous papers. Currently, the examples given in previous papers still serve as a baseline for comparisons, although the numbers w_k have not been recorded. Since we need the w_k as input for our algorithm, we select a learning curve and compute the w_k using the learning curve. The learning curve parameters are taken from the papers where they have been obtained with the least squares method. For the sample data, we compare the maximum likelihood estimates computed using our algorithm with the least squares estimates given in the papers and with the maximum likelihood estimates obtained when using software reliability models other than the hypergeometric reliability model.

2 Hypergeometric model revisited

2.1 Urn model

The hypergeometric software reliability model can be described using an urn model as follows. Suppose that m faults are initially contained in the software. The urn then contains m balls which are coloured black initially. In the first software test, w_1 faults are detected, which corresponds to drawing w_1 balls from the urn. Since the faults detected will be known faults in future tests, the w_1 balls are coloured white and placed back into the urn. In the second test, w_2 faults are detected, which corresponds to drawing w_2 balls from the urn. Some of the faults may be known already, so only x_2 faults are newly detected. This corresponds to having drawn x_2 black balls and $w_2 - x_2$ white balls from the urn. The x_2 black balls are coloured white and all w_2 balls are placed back into the urn. This drawing, colouring white, and replacing of balls is continued until the last test.

According to the urn model, each software test may be modelled probabilistically by a suitable hypergeometric probability distribution.

Definition 1 *For numbers $z \leq m$, $z - c \leq w$, and $w \leq z$, we introduce the notation*

$$p_{m,w}(z, c) = \frac{\binom{m-c}{z-c} \cdot \binom{c}{w-(z-c)}}{\binom{m}{w}}.$$

When m, w, and c are fixed, $p_{m,w}(z, c)$ is a hypergeometric distribution.

Test k is modelled by the hypergeometric distribution

$$p_{m,w_k}(c_k, c_{k-1}).$$

Here, c_k denotes the cumulative number of different faults detected during the first k tests,

$$c_k = x_1 + x_2 + \ \ldots \ + x_k.$$

In previous papers, the hypergeometric distribution for each test has been written down using the number x_k of newly detected faults instead of the cumulative number c_k. Using c_k is equivalent, because $x_k = c_k - c_{k-1}$, and more convenient when defining the stochastic process for the hypergeometric reliability model, see the comment at the end of the next subsection.

The urn model implicitly makes an independence assumption about software faults: it is assumed that detecting a particular fault is stochastically independent of detecting the other faults. This assumption certainly is a simplification, because in practice faults often are related. The independence assumption is accepted as a starting point for software reliability modeling, though. Besides the independence assumption, other assumptions are made in the hypergeometric reliability model which are explicitly stated in the papers, see [12, 19].

2.2 Underlying stochastic process

In the hypergeometric reliability model, each software test is modelled using a hypergeometric distribution. The distribution $p_{m,w_k}(c_k, c_{k-1})$ for test k depends on the outcome of all previous tests, because it has c_{k-1} as a parameter. Therefore, the test series is *not* a Bernoulli sequence. To properly define the stochastic model for the whole test series, two questions must be answered: What is the sample space? How is the probability measure defined on the sample space?

Definition 2 *For a series of n tests, the sample space Ω_n consists of all n-tuples $(c_1, c_2, \ldots c_n)$ of natural numbers*[†] *where the c_k are non-decreasing, $c_k \geq c_{k-1}$.*

For each value of n there is a separate sample space. This has not been made explicit in previous papers.

Definition 3 *Let $\underline{w}$ denote the sequence of numbers $w_1, w_2, \ldots w_n$. Suppose that m and the sequence $\underline{w}$ are given. The probability to observe the outcome $(c_1, c_2, \ldots c_n)$ in the test series is defined as*

$$\mathrm{P}_{m,\underline{w}}(c_1, c_2, \ldots c_n) = p_{m,w_1}(c_1, 0) \cdot p_{m,w_2}(c_2, c_1) \cdot p_{m,w_3}(c_3, c_2) \cdot \ \ldots \ \cdot p_{m,w_n}(c_n, c_{n-1}).$$

Starting from definition 3, it is well-known how to prove by induction on n that the probabilities of all the possible outcomes sum up to one,

$$\sum_{(c_1, c_2, \ldots c_n)} \mathrm{P}_{m,\underline{w}}(c_1, c_2, \ldots c_n) = 1.$$

The proof uses the fact that $p_{m,w}(z, c)$ is a probability distribution for z when m, w, and c are fixed. Thus, the definition yields a *probability measure*

$$\mathrm{P}_{m,\underline{w}}$$

[†] zero included

on the sample space Ω_n.

In the stochastic model just defined, the w_k are parameters and *not* part of the outcome of the random experiment, although in practice it is not known in advance how many faults will be detected in the next test. For each set of parameters m, w_1, w_2, $\dots$ w_n there is a separate probability measure on the sample space. Strictly speaking, the hypergeometric reliability model thus consists of a whole family of probability measures for a given sample space Ω_n. Again, this has not been made explicit in previous papers.

The function C_k on the sample space which yields the cumulative number c_k of different faults detected in the first k tests is a random variable. By construction of the probability measures, the probability to observe a cumulative number of z faults after test k depends on c_{k-1}, but not on the earlier observations c_{k-2}, c_{k-3}, and so on. This is an important property of the random variables C_k:

Proposition 1 *For each probability measure* $\mathrm{P}_{m,\underline{w}}$ *on the sample space* Ω_n*, the stochastic process*

$$C_1, \ C_2, \ \dots \ C_n$$

underlying the hypergeometric software reliability model is a Markov chain.

Proof. We must check that

$$\mathrm{P}_{m,\underline{w}}\,(C_k = z \mid C_{k-1} = c_{k-1}, \ \dots \ C_1 = c_1) \ = \ \mathrm{P}_{m,\underline{w}}\,(C_k = z \mid C_{k-1} = c_{k-1}).$$

By the definition of conditional probabilities, the probability on the left hand side equals

$$\frac{\mathrm{P}_{m,\underline{w}}\,(c_1, \ c_2, \ \dots \ c_{k-1}, \ z)}{\mathrm{P}_{m,\underline{w}}\,(c_1, \ c_2, \ \dots \ c_{k-1})}$$

and the probability on the right hand side equals

$$\frac{\displaystyle\sum_{(c_1, \ c_2, \ \dots \ c_{k-2})} \mathrm{P}_{m,\underline{w}}\,(c_1, \ c_2, \ \dots \ c_{k-1}, \ z)}{\displaystyle\sum_{(c_1, \ c_2, \ \dots \ c_{k-2})} \mathrm{P}_{m,\underline{w}}\,(c_1, \ c_2, \ \dots \ c_{k-1})}.$$

By definition of the probability measure (definition 3), both fractions are equal to $p_{m,w_k}\,(z\,, \ c_{k-1})$. □

The function X_k on the sample space which yields the number $x_k = c_k - c_{k-1}$ of faults newly detected in test k is a random variable, too. The probability to observe x new faults in test k depends on c_{k-1}, that is, on all earlier observations x_{k-1}, x_{k-2}, and so on. Thus, the stochastic process X_1, X_2, $\dots$ X_n is *not* a Markov chain. This is the reason why using the x_k to define the probability space for the hypergeometric software reliability model is less convenient than using the c_k.

2.3 Recursion formula

To estimate the number m of faults initially contained in the software, a least squares method has been used with the hypergeometric reliability model. The least squares method is based on a recursion formula for the expected values $\mathrm{E}_{m,\underline{w}}\,[C_k]$ of the random variables C_k. Note that all expected values in the hypergeometric reliability model depend on the parameters m and $\underline{w}$. The recursion formula has been stated properly (except for the subscripts m and $\underline{w}$) in equation (2) of [12], but it has not been derived properly yet. For example, equation (4) in [19] does not make sense † and the argument in [19] aiming at the recursion formula is incomplete. Using the probability measure $\mathrm{P}_{m,\underline{w}}$ defined above, we prove rigorously that the recursion formula holds:

Proposition 2 *For each probability measure* $\mathrm{P}_{m,\underline{w}}$ *the following recursion formula holds:*

$$\mathrm{E}_{m,\underline{w}}\,[C_k] \ = \ w_k \ + \ (1 \ - \ \frac{w_k}{m}) \ \cdot \ \mathrm{E}_{m,\underline{w}}\,[C_{k-1}].$$

Proof. We suppress the subscripts m and $\underline{w}$ to simplify the notation. The formula for the mean of the hypergeometric distribution [3, 17] yields

$$\mathrm{E}\,[C_k \mid C_{k-1} = c] \ = \ w_k \ + \ (1 \ - \ \frac{w_k}{m}) \ \cdot \ c.$$

This is a formula for the *conditional* expectation of the random variable C_k under the condition that the random variable C_{k-1} assumes the value c. Previous papers have been imprecise about the distinction between expectations and conditional expectations. From the general properties of conditional expectations (see [17]) we get

$$\mathrm{E}\,[C_k] \ = \ \sum_c \mathrm{E}\,[C_k \mid C_{k-1} = c] \ \cdot \ \mathrm{P}\,(C_{k-1} = c).$$

Combine the two equations to compute

$$\mathrm{E}\,[C_k] \ = \ \sum_c \Big(w_k \ + \ (1 \ - \ \frac{w_k}{m}) \cdot c \Big) \ \cdot \ \mathrm{P}\,(C_{k-1} = c)$$

† $\overline{N(i)}$ on the left hand side of equation (4) in [19] is a (conditional) expectation and $C(i - 1)$ on the right hand side is a random variable.

$$= \quad w_k \cdot \sum_c \mathrm{P}(C_{k-1} = c) \; + $$
$$(1 - \frac{w_k}{m}) \cdot \sum_c c \cdot \mathrm{P}(C_{k-1} = c)$$
$$= \; w_k \cdot 1 \; + \; (1 - \frac{w_k}{m}) \cdot \mathrm{E}[C_{k-1}].$$

□

The least squares method searches for a value of m which minimizes the least squares sum

$$\sum_{k=1}^{n} (\mathrm{E}_{m,\underline{w}}[C_k] - c_k)^2.$$

To find a minimizing m for a given test series outcome $c_1, c_2, \ldots c_n$, exhaustive search is performed in [19]. To save computing time, [21] proposes to compute the zeros of the derivative with respect to m of the least squares sum. Since equation (6) in [21] is derived from the incorrect equation (4) in [19], the formula given for the zero of the derivative is incorrect. In [14], genetic algorithms are applied to minimize the least squares sum. The question whether a least squares estimate exists and is unique for each parameter set $\underline{w}$ and test series outcome $c_1, c_2, \ldots c_n$ has not been addressed in the papers.

3 Max likelihood estimates

3.1 Likelihood function

In the stochastic model for the hypergeometric software reliability model, the result of test k consists of the number w_k of faults detected in the test and the cumulative number c_k of different faults detected in the first k tests. The numbers $c_1, c_2, \ldots c_n$ are regarded as the outcome of the random experiment, whereas the numbers $w_1, w_2, \ldots w_n$ are regarded as parameters of the stochastic model. If the parameters w_k are fixed, we get a family of probability measures $\mathrm{P}_{m,\underline{w}}$ on the sample space Ω_n which is parameterized by the number m of faults initially contained in the software. When the outcome $c_1, c_2, \ldots c_n$ is given, the *maximum likelihood estimate for* m answers the question: Which measure $\mathrm{P}_{m,\underline{w}}$ gives the highest probability of observing that particular test series outcome?

Definition 4 *Suppose that a test series outcome $c_1, c_2, \ldots c_n$ and the parameters $w_1, w_2, \ldots w_n$ are given. The likelihood function for the hypergeometric reliability model is defined as*

$$L(m) \; = \; \mathrm{P}_{m,\underline{w}}(c_1, c_2, \ldots c_n).$$

The likelihood function yields for each value of m the probability to observe the given test series outcome in the measure $\mathrm{P}_{m,\underline{w}}$. The likelihood function depends on the observed outcome $c_1, c_2, \ldots c_n$ and the parameters $w_1, w_2, \ldots w_n$. We suppress the corresponding subscripts to the likelihood function to simplify the notation. The likelihood function for the hypergeometric reliability model has been stated in equation (4) of [21] for the special case that the w_k are all the same.

Definition 5 *Suppose that a test series outcome $c_1, c_2, \ldots c_n$ and the parameters $w_1, w_2, \ldots w_n$ are given. The number $\widehat{m}$ is called a maximum likelihood estimate in the hypergeometric reliability model if it satisfies for all m the inequality*

$$L(\widehat{m}) \; \geq \; L(m).$$

A maximum likelihood estimate depends on the observed test series outcome $c_1, c_2, \ldots c_n$ and the parameters $w_1, w_2, \ldots w_n$. We suppress the corresponding subscripts to the maximum likelihood estimate to simplify the notation.

It is not at all obvious that a maximum likelihood estimate exists in the hypergeometric reliability model for a given test series outcome and parameter set. We shall see in subsections 3.3 and 3.4 that for each but one special kind of test series a maximum likelihood estimate exists and can be computed fast.

For the remainder of this section, the parameter set $w_1, w_2, \ldots w_n$ and the test series outcome $c_1, c_2, \ldots c_n$ are assumed to be fixed.

3.2 Growth quotient

To compute a maximum likelihood estimate for a given test series outcome, we must find a global maximum of the likelihood function. In other words, we must study the *growth bevavior* of the likelihood function. Instead of computing the derivative of the logarithm of the likelihood function with respect to m and solving for the zeros of the maximum likelihood equation, we use an elementary approach to study the growth behavior of the likelihood function.

Definition 6 *Define the growth quotient to be*

$$Q(m) \; = \; \frac{L(m)}{L(m-1)}.$$

Clearly, the likelihood function $L(m)$ is increasing when the growth quotient $Q(m)$ is greater than one, and decreasing otherwise. As will be seen in the next two propositions, using the growth quotient has the great advantage that the binomials in the formula for the likelihood function cancel almost completely.

Definition 7 *For $m \neq z$ and $m \neq 0$, we introduce the notation*

$$q_{m,w}(z, c) = \frac{(m-c)\cdot(m-w)}{m\cdot(m-z)}.$$

Proposition 3 *The growth quotient $Q(m)$ admits the presentation*

$$Q(m) = q_{m,w_1}(c_1, 0)\cdot q_{m,w_2}(c_2, c_1)\cdot q_{m,w_3}(c_3, c_2)\cdot \ldots \cdot q_{m,w_n}(c_n, c_{n-1}).$$

Proof. Refer to the definition of $L(m)$ and check

$$\frac{p_{m,w}(z, c)}{p_{m-1,w}(z, c)} = q_{m,w}(z, c).$$

□

The proposition shows that the growth quotient $Q(m)$ is a *rational* function with a nice presentation: both the numerator and the denominator are given as a product of linear factors. At least half of the factors cancel:

Proposition 4 *The growth quotient $Q(m)$ admits the "cancelled" presentation*

$$Q(m) = \frac{(m-w_1)\cdot(m-w_2)\cdot \ldots \cdot(m-w_n)}{m^{n-1}\cdot(m-c_n)}.$$

In addition, for each $w_j = 0$ a factor m cancels. If $c_n = w_k$ for some k, the factor $m - c_n$ cancels.

Proof. Refer to definition 7 and compute

$$q_{m,w_k}(c_k, c_{k-1})\cdot q_{m,w_{k+1}}(c_{k+1}, c_k).$$

□

The proposition shows that the degrees of the polynomials in the numerator and denominator of the growth quotient $Q(m)$ are bounded by the length n of the test series. More important, the proposition shows that the order in which faults are detected as well as the numbers x_k of new faults detected in the tests do not matter for the value of the maximum likelihood estimate: only the sum c_n of different faults and the sizes w_k of the fault samples enter the formula for the growth quotient.

3.3 Growth behavior

The number m of faults initially contained in a software must be greater or equal to the total number c_n of different faults detected during the test series. Therefore, to compute a maximum likelihood estimate it suffices to study the behavior of the growth quotient for

$$m > c_n.$$

Viewed as a function $Q(x)$ of a *real-valued* variable x instead of the integer-valued m, the growth quotient is differentiable and defined for $x \neq c_n$ and $x \neq 0$, see proposition 4. Since the likelihood function $L(m)$ is increasing if and only if the growth quotient is greater than one, we must answer the following question:

> *At which points $x > c_n$ does the graph of the growth quotient $Q(x)$ cross the line $y = 1$?*

Refering to the cancelled presentation of the growth quotient, for $x > c_n$ and test series with $c_n > w_k$ for all k, the growth quotient $Q(x)$ is a product of one or more factors

$$\frac{x - w_j}{x} < 1$$

and the factor

$$\frac{x - w_n}{x - c_n} > 1.$$

Therefore, it is not obvious how the graph of the growth quotient $Q(x)$ is located relative to the line $y = 1$ when $x > c_n$.

Clearly, the growth quotient approaches $y = 1$ as x grows large. Except for some special cases which we shall discuss in the next subsection, the growth quotient has the following properties:

- P1. The graph of the growth quotient $Q(x)$ runs above the line $y = 1$ near c_n.
- P2. Beyond a certain $x_0 > c_n$, the graph of the growth quotient runs below the line $y = 1$.
- P3. Beyond c_n, the graph of the growth quotient has only a single local extreme point, which is a local minimum.

The proofs of these properties are elementary, but too long to be included in this paper. Please refer to [16] for the precise formulation of the theorems and the full proofs.

The properties P1, P2, and P3 determine the behavior of the growth quotient completely for $x > c_n$. By the intermediate-value theorem for continuous functions [1], the first and second property imply that the graph of the growth quotient must cross the line $y = 1$

at least once. Clearly, the graph can cross or touch the line only finitely many times since $Q(x) - 1$ is a rational function. Suppose now that the graph crosses the line more than once. Then there must be (at least) two crossing points such that the graph runs above the line in between. It follows that the graph has a local maximum between these two crossing points, which contradicts the third property. If the graph touches the line at some point without crossing it, that point is a local extreme point (min or max), which again contradicts the third property. To sum up, for most test series we get the following answer to the question raised above:

For $x > c_n$, the graph of the growth quotient $Q(x)$ crosses the line $y = 1$ only once and runs below the line afterwards.

A typical graph of the growth quotient $Q(x)$ for $x > c_n$ is shown in figure 1. In that example, $c_n = 46$.

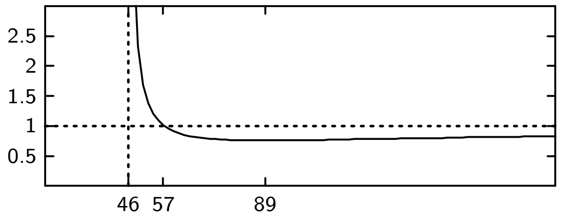

Figure 1: a typical graph of the growth quotient

For the example shown in figure 1, $Q(57) > 1$ whereas $Q(58) < 1$. It follows that the maximum likelihood estimate is $\widehat{m} = 57$. The local minimum of the growth quotient is located between 89 and 90.

3.4 Special cases

There are some special kinds of test series for which the growth quotient $Q(x)$ shows a behavior different from that described by figure 1. The special cases are:

Case A. $c_n = w_k$ for some index k and $w_j > 0$ for at least one index $j \neq k$;

Case B. $c_n = w_k$ for some index k, but $w_j = 0$ for all $j \neq k$;

Case C. $c_n > w_k$ and $w_k = c_k - c_{k-1}$ for all k.

Cases A and B may be interpreted as follows. The condition $c_n = w_k$ means that the effort for the first $k - 1$ and the last $n - k$ tests could have been saved, because test k would have sufficed to detect all c_n faults which are known by the end of the full test series. For case A, we show in [16] that

$$Q(m) < 1$$

for all $m > c_n$. Thus, the maximum likelihood estimate is unique and equals

$$\widehat{m} = c_n$$

in case A.

As compared to case A, in case B faults are detected *only* in test k. For case B, we show in [16] that

$$Q(m) = 1$$

for all $m > c_n$. Thus,

$$\text{any } m \geq c_n$$

is a maximum likelihood estimate in case B.

Case C may be explained as follows. The condition $w_k = c_k - c_{k-1}$ for all k means that in each test all detected faults are new. This outcome indicates a high number of faults in the software. In the hypergeometric reliability model, such an outcome is the more likely the larger the number of faults contained in the software is, and we show in [16] that

$$Q(m) > 1$$

for all $m > c_n$. Thus,

$$\text{no maximum likelihood estimate}$$

exists in case C.

All three special cases are unlikely to occur in a software test series. Typically, no single test will uncover *all* faults known to be in the software by the end of the test series (cases A and B). Also, faults will be detected in more than one test (case B) and some faults will be re-detected in later tests of a series (case C). Thus, the special cases are unrealistic and can be disregarded in software testing practice.

4 Algorithm

The behavior of the growth quotient $Q(x)$ in the typical case implies that the likelihood function $L(m)$ is increasing *only until* the graph of the growth quotient falls below the line $y = 1$ for the first time, see figure 1. This fact gives us a *fast algorithm* for computing the maximum likelihood estimate in the hypergeometric reliability model:

Step 1. Check whether one of the three special cases applies, see subsection 3.4. If no, continue with Step 2.

Step 2. Compute the maximum likelihood estimate $\widehat{m}$ iteratively as follows.

Step 2a. Set $x := c_n + 1$.

Step 2b. Check whether $Q(x) > 1$. If yes, set $x := x + 1$ and repeat Step 2b. If no, return the maximum likelihood estimate $\widehat{m} = x - 1$.

The growth quotient $Q(x)$ is a rational function. The polynomials in the numerator and denominator of the growth quotient are of degree n, where n equals the number of tests in the series, or of a lower degree if some $w_j = 0$. Typically, the number of tests will be large. Thus, the growth quotient must be computed in Step 2b of the algorithm in a way which avoids problems with the precision of the floating point arithmetic. For example, one might take advantage of the "cancelled" presentation of the growth quotient given in Proposition 4. Re-order the w_k in increasing order according to their value, so that w_n is largest. Compute

$$\frac{x - w_n}{x - c_n}$$

and multiply with the factors

$$\frac{x - w_j}{x}$$

until the product is less than one, or, all w_j have been used up. When computing $Q(x)$ this way, the factors have a similar order of magnitude.

5 Examples

We use our algorithm to compute the maximum likelihood estimates for two examples taken from previous papers. In these examples, the numbers w_k have not been recorded. Since we need the w_k as input for our algorithm, we select a learning curve and compute the w_k using the learning curve. The learning curve parameters are taken from the papers, where they have been obtained with the least squares method.

The first example is taken from [19, 21]. In the example, a total of $c_n = 481$ different faults were detected in a series of $n = 111$ tests. For each test, the number x_k of new faults and the number u_k of testers involved in the test were recorded. The number w_k is computed using the learning curve

$$w_k = u_k \cdot (a \cdot k + b).$$

The learning curve parameters a and b are listed in Table V of [21] as

$$a = 0.082 \quad \text{and} \quad b = 1.36.$$

Several adjustments had to be made to the computed values for the w_k which seem to have been overlooked in previous papers:

- All computed values were rounded up to the next integer.
- All faults detected in the first test are new. Thus, we adjusted w_1 from 6 to 5.
- For any test, the number w of faults detected must be greater or equal to the number x of new faults. Thus, we adjusted

w_{11}	from	12	to	31;
w_{13}	from	13	to	24;
w_{14}	from	13	to	49;
w_{15}	from	13	to	14.

The final values for the w_k are listed in table 1 of the appendix.

The graph of the first example's growth quotient $Q(x)$ is shown for $x > c_n$ in figure 2.

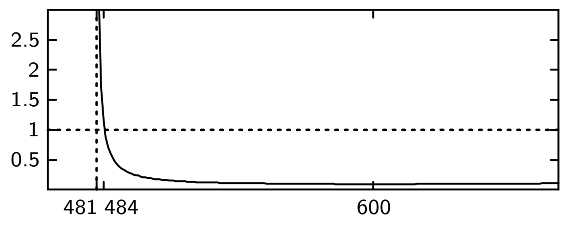

Figure 2: $Q(x)$ for the first example

For the first example,

$$Q(484) = 1.159 \quad \text{and} \quad Q(485) = 0.880.$$

Since $Q(484) > 1$ whereas $Q(485) < 1$, the maximum likelihood estimate is

$$\widehat{m} = 484.$$

The local minimum of the growth quotient is located between 600 and 601. The computation of the maximum likelihood estimate took less than a second on a standard personal computer. The least squares estimate given in [21] for the example is equal to 484, too.

The second example is taken from [6]. In the example, a total of $c_n = 328$ different faults were detected in a series of $n = 19$ tests. For each test, the number x_k of new faults and the execution time t_k of the test in milliseconds were recorded. The number w_k is computed using the learning curve

$$w_k = m \cdot \frac{t_k}{1000} \cdot (a \cdot k + b).$$

Note that the learning curve is parameterized by the estimate $m = 387.71$ for the number of faults initially contained in the software, which has been computed together with the learning curve parameters in [6] using the least squares method. We have chosen this example nonetheless because we wanted to compare the maximum likelihood estimate obtained in the hypergeometric reliability model with the maximum likelihood estimates obtained in other reliability models, see below.

The parameters a and b are listed in Table 3 of [6],

$$a = 0.0017 \quad \text{and} \quad b = 0.0224.$$

As in the first example, several adjustments had to be made to the computed values for the w_k. We adjusted

$$\begin{array}{llllll} w_1 & \text{from} & 23 & \text{to} & 15; \\ w_2 & \text{from} & 25 & \text{to} & 29; \\ w_3 & \text{from} & 21 & \text{to} & 22; \\ w_4 & \text{from} & 12 & \text{to} & 37. \end{array}$$

The final values for the w_k are listed in table 2 of the appendix.

The graph of the second example's growth quotient $Q(x)$ is shown for $x > c_n$ in figure 3.

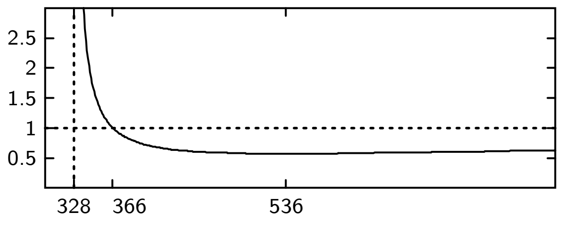

Figure 3: $Q(x)$ for the second example

For the second example,

$$Q(366) = 1.0089 \quad \text{and} \quad Q(367) = 0.9922.$$

It follows that the maximum likelihood estimate is

$$\widehat{m} = 366.$$

The local minimum of the growth quotient is located between 536 and 537. The least squares estimate given in [6] for the example is equal to 388.

Finally, we compare the maximum likelihood estimate obtained in the hypergeometric reliability model for the second example with the maximum likelihood estimates obtained in other software reliability models, see Table 4 in [6]:

model	estimate $\widehat{m}$
exponential [4]	455
delayed S-shaped [22]	351
inflection S-shaped [15]	347
hypergeometric	366

The maximum likelihood estimate in the hypergeometric reliability model is much closer to the estimates in the S-shaped models than to the estimate in the exponential model.

6 Added in proof

The formula from definition 3 in subsection 2.2 of this paper is also given in subsection 4.3.3 of the book [2]. The question of the existence and uniqueness of maximum likelihood estimates for the hypergeometric model is also addressed in [2], but the special case B from subsection 3.2 of this paper is missing in [2].

References

1. T. M. Apostol: *Mathematical Analysis.* Addison-Wesley 1974
2. K.-Y. Cai: *Software Defect and Operational Profile Modeling.* Kluwer 1998
3. W. Feller: *An Introduction to Probability Theory and Its Applications.* Wiley 1968
4. A. L. Goel, K. Okumoto: "Time-Dependent Error-Detection Rate Model for Software Reliability and Other Performance Measures", IEEE Transactions on Reliability 28:3 (1979) 206-211
5. R.-H. Hou, I.-Y. Chen, Y.-P. Chang, S.-Y. Kuo: "Optimal Release Policies for Hyper-Geometric Distribution Software Reliability Growth Model with Scheduled Delivery Time", Proceedings Asia-Pacific Software Engineering Conference APSEC (1994) 445-452
6. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Applying Various Learning Curves to Hyper-Geometric Distribution Software Reliability Growth Model", Proceedings International Symposium on Software Reliability Engineering ISSRE 5 (1994) 8-17
7. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Hyper-Geometric Distribution Software Reliability Growth Model with Imperfect Debugging", Proceedings International Symposium on Software Reliability Engineering ISSRE 6 (1995) 195-200
8. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Efficient Allocation of Testing Resources for Software Module Testing Based on the Hyper-Geometric Distribution Software Reliability Growth Model", Proceedings International Symposium on Software Reliability Engineering ISSRE 7 (1996) 289-298
9. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Needed Resources for Software Module Test, Using the Hyper-Geometric Software Reliability Growth Model", IEEE Transactions on Reliability 45:4 (1996) 541-549
10. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Optimal Release Policy for Hyper-Geometric Distribution Software-Reliability Growth Model", IEEE Transactions on Reliability 45:4 (1996) 646-651

11. R.-H. Hou, S.-Y. Kuo, Y.-P. Chang: "Optimal Release Times for Software Systems with Scheduled Delivery Time Based on HGDM", IEEE Transactions on Computers 46:2 (1997) 216-221

12. R. Jacoby, Y. Tohma: "The Hypergeometric Distribution Software Reliability Growth Model: Precise Formulation and Applicability", Proceedings International Computer Software and Applications Conference COMPSAC (1990) 13-19

13. R. Jacoby, Y. Tohma: "Parameter Value Computation by Least Squares Method and Evaluation of Software Availability and Reliability at Service-Operation by the Hyper-Geometric Distribution Software Reliability Growth Model", Proceedings International Conference on Software Engineering ICSE 13 (1991) 226-237

14. T. Minohara, Y. Tohma: "Parameter Estimation of Hyper-Geometric Distribution Software Reliability Growth Model by Genetic Algorithms", Proceedings International Symposium on Software Reliability Engineering ISSRE 6 (1995) 324-329

15. M. Ohba: "Software Reliability Analysis Models", IBM Journal of Research and Development 28:4 (1984) 428-443

16. F. Padberg: "Maximum Likelihood Estimates for the Hypergeometric Software Reliability Model", Technical Report, Universität Karlsruhe, Germany, 2001

17. E. Parzen: *Modern Probability Theory and Its Applications*. Wiley 1960

18. Y. Tohma, R. Jacoby, Y. Murata, M. Yamamoto: "Hypergeometric Distribution Model to Estimate the Number of Residual Software Faults", Proceedings International Computer Software and Applications Conference COMPSAC (1989) 610-617

19. Y. Tohma, K. Tokunaga, S. Nagase, Y. Murata: "Structural Approach to the Estimation of the Number of Residual Software Faults Based on the Hypergeometric Distribution", IEEE Transactions on Software Engineering 15:3 (1989) 345-355

20. Y. Tohma, H. Yamano, M. Ohba, R. Jacoby: "Parameter Estimation of the Hyper-Geometric Distribution Model for Real Test/Debug Data", Proceedings International Symposium on Software Reliability Engineering ISSRE 2 (1991) 28-34

21. Y. Tohma, H. Yamano, M. Ohba, R. Jacoby: "The Estimation of Parameters of the Hypergeometric Distribution and Its Application to the Software Reliability Growth Model", IEEE Transactions on Software Engineering 17:5 (1991) 483-489

22. S. Yamada, M. Ohba, S. Osaki: "S-Shaped Reliability Growth Modeling for Software Error Detection", IEEE Transactions on Reliability 32:5 (1983) 475-485

Appendix

k	x_k	u_k	w_k
1	5	4	5
2	5	4	7
3	5	4	7
4	5	4	7
5	6	4	8
6	8	5	10
7	2	5	10
8	7	5	11
9	4	5	11
10	2	5	11
11	31	5	31
12	4	5	12
13	24	5	24
14	49	5	49
15	14	5	14
16	12	5	14
17	8	5	14
18	9	5	15
19	4	5	15
20	7	5	15
21	6	5	16
22	9	5	16
23	4	5	17
24	4	5	17
25	2	5	18
26	4	5	18
27	3	5	18
28	9	6	22
29	2	6	23
30	5	6	23
31	4	6	24
32	1	6	24
33	4	6	25
34	3	6	25
35	6	6	26
36	13	6	26
37	19	8	36
38	15	8	36
39	7	8	37
40	15	8	38
41	21	8	38
42	8	8	39
43	6	8	40
44	20	8	40
45	10	8	41
46	3	8	42
47	3	8	42
48	8	4	22
49	5	4	22
50	1	4	22
51	2	4	23
52	2	4	23
53	2	4	23
54	7	4	24
55	2	4	24
56	0	4	24
57	2	4	25
58	3	4	25
59	2	4	25
60	7	4	26
61	3	4	26
62	0	4	26
63	1	4	27
64	0	4	27
65	1	4	27
66	0	3	21
67	0	3	21
68	1	3	21
69	1	3	22
70	0	3	22
71	0	3	22
72	1	3	22
73	1	4	30
74	0	4	30
75	0	4	31
76	0	4	31
77	1	4	31
78	2	2	16
79	0	2	16
80	1	2	16
81	0	2	17
82	0	2	17
83	0	2	17
84	0	2	17
85	0	2	17
86	0	2	17
87	2	2	17
88	0	2	18
89	0	2	18
90	0	2	18
91	0	2	18
92	0	2	18
93	0	2	18
94	0	2	19
95	0	2	19
96	1	2	19
97	0	2	19
98	0	2	19
99	0	2	19
100	1	2	20
101	0	1	10
102	0	1	10
103	1	1	10
104	0	1	10
105	0	1	10
106	1	1	11
107	0	1	11
108	0	1	11
109	1	1	11
110	0	1	11
111	1	1	11

Table 1: input data for the first example

k	x_k	t_k	w_k
1	15	2450	15
2	29	2450	29
3	22	1960	22
4	37	980	37
5	2	1680	21
6	5	3370	43
7	36	4210	56
8	29	3370	48
9	4	960	15
10	27	1920	30
11	27	2880	46
12	22	1440	24
13	21	3260	57
14	22	3840	69
15	6	3840	72
16	7	2300	45
17	9	1760	36
18	5	1990	41
19	3	2990	64

Table 2: input data for the second example