Performance Optimized Expectation Conditional Maximization Algorithms for Nonhomogeneous Poisson Process Software Reliability Models

Vidhyashree Nagaraju, *Member, IEEE*, Lance Fiondella, *Member, IEEE*, Panlop Zeephongsekul, *Member, IEEE*, Chathuri L. Jayasinghe, and Thierry Wandji, *Member, IEEE*

***Abstract*—Nonhomogeneous Poisson process (NHPP) and software reliability growth models (SRGM) are a popular approach to estimate useful metrics such as the number of faults remaining, failure rate, and reliability, which is defined as the probability of failure free operation in a specified environment for a specified period of time. We propose performance-optimized expectation conditional maximization (ECM) algorithms for NHPP SRGM. In contrast to the expectation maximization (EM) algorithm, the ECM algorithm reduces the maximum-likelihood estimation process to multiple simpler conditional maximization (CM)-steps. The advantage of these CM-steps is that they only need to consider one variable at a time, enabling implicit solutions to update rules when a closed form equation is not available for a model parameter. We compare the performance of our ECM algorithms to previous EM and ECM algorithms on many datasets from the research literature. Our results indicate that our ECM algorithms achieve two orders of magnitude speed up over the EM and ECM algorithms of [1] when their experimental methodology is considered and three orders of magnitude when knowledge of the maximum-likelihood estimation is removed, whereas our approach is as much as 60 times faster than the EM algorithms of [2]. We subsequently propose a two-stage algorithm to further accelerate performance.**

***Index Terms*—Expectation conditional maximization (ECM) algorithm, nonhomogeneous Poisson process (NHPP), software reliability, software reliability growth model, two-stage algorithm.**

NOMENCLATURE

Acronyms

EM	Expectation-maximization.
ECM	Expectation conditional maximization.
DSS	Delayed S-shaped model.
CDF	Cumulative distribution function.
LL	Log-likelihood function.
RLL	Reduced log-likelihood function.
MLE	Maximum-likelihood estimation.
MVF	Mean value function.
NHPP	Nonhomogeneous Poisson process.
SRGM	Software reliability growth models.
GO	Goel–Okumoto model.
ISS	Inflexion S-shaped model.

Notation

$m(t)$	MVF of NHPP.
$\lambda(t)$	Instantaneous failure rate.
$F(t)$	Cumulative distribution function of software fault detection process.
a	Number of latent faults at start of testing.
b	Scale parameter of Weibull SRGM testing.
c	Shape parameter of Weibull SRGM testing.
ϕ	Rate at which bathtub transitions. from second to final phase.
Ψ	Inflexion parameter.
r	Inflexion rate.
Γ	Gamma function.
$\digamma$	Polygamma function.
G	MeijerG function.
N	Total number of faults.
n	Observed number of faults.
m	Unobserved number of faults.
$\mathbf{T}$	Vector of failure times.
t_i	Time of the ith failure.
t_n	nth observed failure.
t_{obs}	Time at which testing stopped.
s	Mission time.
x	Observed data.
z	Unobserved data.
Θ	Vector of model parameters.
p	Number of model parameters.
Δ	Vector of conditional maximization steps.
δ	Subvector of conditional maximization steps.
$\Theta_i^{(j)}$	ith model parameter in the jth iteration

Manuscript received May 10, 2016; revised November 3, 2016 and January 7, 2017; accepted June 12, 2017. Date of publication July 6, 2017; date of current version August 30, 2017. This work was supported in part by the Naval Air Systems Command through the Systems Engineering Research Center, a Department of Defense University Affiliated Research Center under Research Task 139: Software Reliability Modeling and in part by the National Science Foundation under Grant 1526128. Associate Editor: Z. Chen. *(Corresponding author: Lance Fiondella.)*

V. Nagaraju and L. Fiondella are with the Department of Electrical and Computer Engineering, University of Massachusetts, Dartmouth, MA 02747 USA (e-mail: vnagaraju@umassd.edu; lfiondella@umassd.edu).

P. Zeephongsekul is with the School of Mathematical and Geospatial Sciences, RMIT University, Melbourne, Vic 3000, Australia (e-mail: panlopz@rmit.edu.au).

C. L. Jayasinghe is with the Department of Statistics, Faculty of Applied Sciences, University of Sri Jayewardenepura, Nugegoda 10250, Sri Lanka (e-mail: chathuri@sjp.ac.lk).

T. Wandji is with the Naval Air Systems Command, Patuxent River, MD 20670 USA (e-mail: ketchiozo.wandji@navy.mil).

Digital Object Identifier 10.1109/TR.2017.2716419

I. Introduction

SOFTWARE reliability [3], commonly defined as the probability of failure free operation in a specified environment for a specified period of time, can be estimated with the assistance of software reliability growth models. SRGMs [4], [5] are a fundamental and well-established methodology, many of which are based on the NHPP [6], [7]. These NHPP SRGM [8] also enable the estimation of useful metrics such as prediction of the number of faults remaining, failure intensity, etc., SRGM are also used in optimization problems [9] to determine the amount of testing required to achieve a desired level of reliability [10] and to minimize testing costs, while considering the risk of post release failures [11].

A significant challenge associated with SRGM is the complexity of estimating the parameters of a model [12] with traditional fitting procedures that perform MLE [13]. This difficulty arises because traditional numerical procedures to find the MLEs of a software failure dataset such as the Newton–Raphson method [14] are sensitive to initial parameter estimates and can fail to converge to the MLE if the initial parameter estimates are not sufficiently close to the MLE. This sensitivity of existing model fitting procedures requires a relatively high level of experience, which is the primary reason why many potential users are deterred from applying NHPP-based SRGM to quantitatively assess the reliability of their software. Given the increasing demand for reliable software, a model fitting procedure that is less sensitive to initial parameter estimates is needed so that software reliability growth models can be fit to data with relatively little or no effort. Such a procedure will simplify the application of NHPP-based SRGM and encourage their widespread use if an algorithm exhibiting a combination of stability and performance is incorporated into an automated tool.

Our previous research [1] developed EM [15] and ECM [16] algorithms for NHPP SRGM using the theory of stochastic point processes to derive CM-steps for failure time and failure count variants of the Weibull and Gamma models. The EM and ECM algorithms exhibit greater stability than the Newton–Raphson method. However, the CM-steps obtained through the stochastic point process approach contain semiinfinite integrals and several special functions, which must be computed in each iteration of the algorithms; thus, slowing performance. To overcome this shortcoming, this paper presents performance optimized ECM algorithms to identify the MLEs of the parameters of an NHPP SRGM. The numerical nature of our ECM algorithm simplifies the maximum-likelihood estimation process by reducing a p-dimensional problem to p one-dimensional (1-D) problems, drastically simplifying the computation. Two variants of the ECM are proposed. The first derives CM-steps for all parameters of an SRGM, whereas the second approach eliminates one parameter from the estimation process, reducing the number of model parameters by one. We apply these algorithms to several combinations of well-known models [17]–[22] and datasets [4], [23] from the literature to assess their performance. Our contributions are twofold:

1) The proposed ECM algorithms achieve two orders of magnitude speed up over the EM and ECM algorithms of [1] when their experimental methodology is considered and three orders of magnitude when knowledge of the MLEs is removed, whereas our approach is as much as 60 times faster than the EM algorithms of [2].
2) We subsequently accelerate our algorithm by an additional order of magnitude by combining our ECM algorithm into a two-stage algorithm that first performs a finite number of ECM iterations, but stops prior to convergence and then provides these intermediate parameter estimates as input Newton's Method, achieving even faster convergence.

The remainder of the paper is organized as follows. Section II provides an overview of software reliability growth models. Section III reviews methods to estimate the parameters of an SRGM, including maximum-likelihood estimation as well as expectation–maximization and ECM algorithms. Section IV presents performance optimized ECM algorithms for NHPP SRGM. Section V illustrates the effectiveness and performance of these algorithms through numerical examples. Section VI offers conclusions and directions for future research.

II. Software Reliability Growth Modeling

This section provides an overview of an NHPP software reliability growth models and then presents several models, including the GO [17], DSS [18], Weibull [19], ISS [20], gamma [21], and bathtub [22].

A. NHPP Software Reliability Growth Models

The nonhomogeneous Poisson process is a stochastic process [6] that counts the number of events that occur by time t. The expected value of an NHPP is characterized by the MVF, denoted $m(t)$. The MVF can take many functional forms. In the context of software reliability, the NHPP counts the number of faults detected after the software has been tested for a given period of time. The MVF of several SRGM can be written as

$$m(t) = a \times F(t) \tag{1}$$

where a denotes the number of faults to be detected with infinite testing and $F(t)$ is the cumulative distribution function (CDF) of a continuous probability distribution, characterizing the software fault detection process.

The rate of occurrence of failures is time varying with instantaneous failure rate

$$\lambda(t) = \frac{\mathrm{d}m(t)}{\mathrm{d}t}. \tag{2}$$

1) GO SRGM: The GO model was originally proposed by Goel and Okumoto [17]. The MVF is

$$m(t) = a(1 - e^{-bt}) \tag{3}$$

where b is the fault detection rate.

2) DSS SRGM: Another SRGM that follows the form of (1) is the DSS model proposed by Yamada *et al.* [18]. The MVF of the DSS SRGM is

$$m(t) = a\left(1 - (1 + bt)e^{-\mathrm{bt}}\right) \tag{4}$$

where the term $bte^{-\mathrm{bt}}$ can characterize a delay in fault detection induced by phenomenon such as delayed failure reporting and fault masking.

3) Weibull SRGM: The MVF of the Weibull model [19] is

$$m(t) = a\left(1 - e^{-\mathrm{b}t^c}\right). \tag{5}$$

Here, b and c are the scale and shape parameters, respectively. Setting $c = 1$ in (5) simplifies to the GO model [11].

4) ISS SRGM: The MVF of the inflection S-shaped model [20] is

$$m(t) = a\frac{1 - e^{-\mathrm{b}t}}{1 + \psi e^{-\mathrm{b}t}}. \tag{6}$$

Here, b is the constant fault detection rate and ψ is the inflection parameter. The inflection parameter [22] is defined as

$$\psi(r) = \frac{1 - r}{r}, r \in (0, 1]. \tag{7}$$

where r is the inflection rate that provides the ratio of the number of detectable faults to the total number of faults in the system due to masking and other causes. As r approaches 1.0, the inflection S-shaped model reduces to the GO model.

5) Gamma SRGM: The MVF of the gamma SRGM [21] is

$$m(t) = a\int_0^t \frac{b^c x^{c-1} e^{-bx}}{\Gamma(c)} dx \tag{8}$$

where the identity $F(t) = \int f(t)dt$ has been used and $\Gamma(c) = \int_0^\infty x^{c-1}e^{-x}dx$ is the gamma function. Here, b and c are the scale and shape parameters. Setting $c = 2$ in (8) simplifies to the DSS model [24].

6) Bathtub SRGM: Another SRGM that follows the form of (1) is the bathtub model [22]. The following distribution [25] is bathtub-shaped when $b > 0$ and $c < 1$

$$\beta(t) = bct^{c-1}e^{\phi t}. \tag{9}$$

Substituting (9) into (1), the MVF of the bathtub SRGM is

$$m(t) = a\left(1 - e^{-bct^c e^{\phi t}}\right) \tag{10}$$

where b characterizes the ease of detecting simple faults when testing commences, c influences the rate at which fault detection approaches the second phase where functional requirements are verified, and ϕ describes the rate at which the bathtub transitions to the third and final phase of the testing process which may be characterized by code comprehension [26]. It can be shown that setting $\phi = 0$ in (9) reduces to the Weibull model. Nevertheless, Fiondella and Gokhale [22] demonstrated that there are datasets where the bathtub model outperforms the Weibull model with respect to both information theoretic and predictive measures of goodness of fit, indicating that all three stages of the bathtub may be present in some software testing processes.

III. Parameter Estimation Methods

This section describes various methods to estimate the parameters of an SRGM with the method of maximum-likelihood estimation, including Newton's method [14] as well as the EM [27] and ECM [16] algorithms.

A. Maximum-Likelihood Estimation and Newton's Method

Maximum-likelihood estimation maximizes the likelihood function, also known as the joint distribution of the failure data. Commonly, the logarithm of the likelihood function is maximized because the monotonicity of the logarithm ensures that the maximum of the LL function is equivalent to maximizing the likelihood function and application of logarithmic identities simplifies the derivation of LL estimates. Observed failure time data consist of a vector of individual failure times $\mathbf{T} = \langle t_1, t_2, \ldots, t_n\rangle$ with density function $f(t_i; \Theta)$. The LL function of a failure times dataset is

$$\mathrm{LL}(t_i; \Theta) = -m(t_n) + \sum_{i=1}^{n} \log\left(\lambda(t_i)\right) \tag{11}$$

where Θ is the vector of model parameters and $\lambda(t_i)$ is the instantaneous failure rate at time t_i. The MLE is found by numerically solving the following system of simultaneous equations:

$$\frac{\partial}{\partial\Theta}\mathrm{LL}(\Theta) = \mathbf{0} \tag{12}$$

with an algorithm such as the Newton–Raphson method [14].

The Newton–Raphson method is a numerical algorithm to identify the roots of an equation. The iterative update rule is

$$\Theta = \Theta' - \mathbf{H}^{-1}(\Theta')\mathbf{u}(\Theta') \tag{13}$$

where Θ' is the vector of present parameter estimates, $\mathbf{H}$ is the Hessian matrix, and $\mathbf{u}$ is the score vector defined in (12). However, the Newton–Raphson method may not converge when the initial estimates chosen as input are not close to the maximum.

Given the parameter estimates of a model, software reliability is defined as the probability of failure free operation in the time interval $(t, t + s)$ [5]

$$R(s, t) = e^{(-\hat{m}(t+s) - \hat{m}(t))} \tag{14}$$

where $\hat{m}(t)$ is the MVF of the model evaluated with the numerical parameter estimates obtained from maximum-likelihood estimation and s is the mission time.

B. EM Algorithm

The EM algorithm computes the MLEs by maximizing with respect to the complete data, which consists of observed and unobserved data. Thus, the EM algorithm maximizes the LL function of the complete data, which can be expressed as

$$E(\ln(f(x, \mathbf{Z}; \Theta))|\Theta', x) = \int \ln(f(x, z; \Theta))f(z|x; \Theta')dz \tag{15}$$

where $x = \langle x_1, x_2, \ldots, x_n\rangle$ is the observed data and $z = \langle z_1, z_2, \ldots, z_m\rangle$ the unobserved data, both possessing probability density $f(\cdot; \Theta)$ and Θ' are the previous iteration's parameter estimates. $\ln(f(x, z; \Theta))$ is the LL function of the complete data $y = (x, z)$ of length $N = n + m$ and $f(z|x; \Theta')$ is the conditional density of the unobserved data.

Let $t_1 < t_2 < \ldots < t_N$ be the fault detection times of the complete data, where N is the total number of faults in the software and a Poisson distributed random variable with parameter $a > 0$. The complete LL function for a SRGM of the form given

in (1) is

$$\mathrm{LL}(a,\Theta) = N\log(a) - a + \sum_{i=1}^{N}\log\left(f(t_i;\Theta)\right). \tag{16}$$

By the first-order optimality condition, initial estimates of parameter a and the additional parameters of the probability distribution function $F(t;\Theta)$ are computed based on the observed data such that

$$a^{(0)} = n \tag{17}$$

and

$$\Theta^{(0)} := \sum_{i=1}^{n}\frac{\partial}{\partial\Theta}\log\left(f(t_i;\Theta)\right) = \mathbf{0}. \tag{18}$$

Okamura *et al.* [27] showed that for a MVF of the form $m(t) = a \times F(t)$, an initial estimate of the number of faults (a) is simply the number of observed faults (n), while the remaining initial parameter estimates can be determined by maximizing the LL function of the probability density function $f(\cdot;\Theta) = 0$ and solving to obtained closed-form expressions for these additional parameters. In cases where (18) lacks a closed form solution for a model parameter, a practical strategy is to explicitly specify an initial feasible value for the parameter in a manner that simplifies the model form. For example, parameter c of the Weibull lacks a closed form. However, setting $c^{(0)} = 1$ simplifies to the GO model, providing an initial feasible estimate for all model parameters.

For example, the initial estimate of the scale parameter of the GO SRGM is obtained from (3) and (18)

$$b^{(0)} = \frac{n}{\sum_{i=1}^{n} t_i}. \tag{19}$$

Similarly, the initial estimate of the scale parameter of the DSS SRGM is obtained from (4) and (18)

$$b^{(0)} = \frac{2n}{\sum_{i=1}^{n} t_i}. \tag{20}$$

Given $\mathbf{T} = \langle t_1,\ldots,t_n,t_{\mathrm{obs}}\rangle$, where $\mathbf{T}$ is incomplete data because all the software faults may not have been detected by the end of testing t_{obs}, the EM algorithm is developed by taking the expected values of (17) and (18) in light of $\mathbf{T}$. This produces $a^{'}$ and $\Theta^{'}$, which are the parameter estimates in iteration j. The model parameter estimates are calculated as

$$a^{''} = E[N|\mathbf{T};a^{'},\Theta^{'}] \tag{21}$$

and

$$\Theta^{''} := E\left[\sum_{i=1}^{N}\frac{\partial}{\partial\Theta}\log\left(f(t_i;\Theta)\right)|\mathbf{T};a^{'},\Theta^{'}\right] = \mathbf{0}. \tag{22}$$

The expected values in (21) and (22) are [27]

$$E\left[\sum_{i=1}^{N} h(t_i)|\mathbf{T};a^{'},\Theta^{'}\right] = \sum_{i=1}^{n} h(t_i) + a^{'}\times\int_{t_{\mathrm{obs}}}^{\infty} h(u)f(u;\Theta^{'})du \tag{23}$$

where $h(\cdot)$ is typically an identity function.

For example, the update rules for the a and b parameters of the GO SRGM are [27]

$$a^{''} := n + a^{'}e^{-b^{'}t_{\mathrm{obs}}} \tag{24}$$

$$b^{''} := \frac{n + a^{'}e^{-b^{'}t_{\mathrm{obs}}}}{\sum_{i=1}^{n} t_i + a^{'}\left(t_{\mathrm{obs}} + \frac{1}{b^{'}}\right)e^{-b^{'}t_{\mathrm{obs}}}} \tag{25}$$

where $a^{'}$ and $b^{'}$ are the estimates of a and b identified in the previous iteration. These update rules are computed iteratively until some convergence criterion [2] is satisfied.

C. ECM Algorithm

This section provides a brief overview of the ECM algorithm. Additional mathematical details are provided in [1]. Unlike the EM algorithm which commonly requires the solution of computationally intensive expressions for complex SRGMs, the ECM algorithm [16] simplifies computation by dividing a single M-step into p conditional-maximization (CM)-steps, where p denotes the number of model parameters. Instead of solving a system of simultaneous equations as a single p-dimensional M-step, the ECM algorithm updates only one parameter at a time holding all others constant, and thus, reduces the maximum-likelihood estimation process to p 1-D problems.

In each CM-step of the ECM algorithm searches a single dimension of the parameter space to improve the LL. This is implemented by partitioning the vector of model parameters Θ into subvectors $\langle\Theta_1,\ldots,\Theta_p\rangle$. Successive CM-steps determine $\Theta_i^{(j)}$, which is the updated value of the ith parameter in the jth iteration. Let

$$\Delta = \{\delta_j(\Theta); j = 1,\ldots,p\}. \tag{26}$$

Without loss of generality, the CM-step which updates the ith parameter in the jth iteration takes $\Theta^{(jp+i)} = \langle\Theta_1^{(j+1)},\Theta_2^{(j+1)},\ldots,\Theta_{i-1}^{(j+1)},\Theta_i^{(j)},\ldots,\Theta_p^{(j)}\rangle$ as input, holds all values but $\Theta_i^{(j)}$ constant, and maximizes the partial derivative of the LL function with respect to Θ_i to produce $\Theta^{jp+(i+1)}$ containing $\Theta_i^{(j+1)}$. Each CM-step improves the LL function monotonically. Thus, the ECM algorithm preserves the monotonicity property of the EM algorithm [15] as noted in [1].

IV. Performance Optimized ECM Algorithm

This section presents the steps of an ECM algorithm which avoids the complexity of the traditional EM and ECM algorithm. Unlike the previous EM [2] and ECM [1] algorithms, we do not seek to obtain closed form expression for model parameters and instead solve the update rules numerically. This approach achieves significantly greater computational efficiency. Here, we describe the steps of a procedure for a reduced LL ECM (RLL-ECM) algorithm, briefly noting how to derive the CM-steps from the LL function.

1) *S.1:* Step one applies (11) to determine the LL function of a failure times NHPP SRGM.
2) *S.2:* Step two reduces the LL function from p to $(p-1)$ parameters by differentiating the LL function with respect

to a, equating the result to zero, and solving for a

$$\frac{\partial \mathrm{LL}}{\partial a} = 0. \tag{27}$$

When the MVF possesses the form $a \times F(t)$, it follows from (11) that

$$\hat{a} = \frac{n}{F(t_n)}. \tag{28}$$

Substituting the solution of (27) or (28) into the LL function produces a reduced LL (RLL) function with $(p-1)$ model parameters.

3) *S.3:* Step three derives the conditional maximum (CM)-steps for the remaining $(p-1)$ parameters by computing partial derivatives

$$\frac{\partial \mathrm{RLL}}{\partial \Theta_i} = 0 \tag{29}$$

for $(1 \le i \le p-1)$.

4) *S.4:* Step four cycles through the $(p-1)$ CM-steps holding the other $(p-2)$ parameters constant at the values from the previous iteration and then applying a numerical root finding algorithm to the single nonconstant parameter. This cycle repeats until a convergence criterion such as

$$|\mathrm{RLL}_j - \mathrm{RLL}_{j-1}| < \varepsilon \tag{30}$$

is satisfied, where $\varepsilon > 0$ is an arbitrarily small constant. This identifies the MLEs of all parameters but a, denoted $\widehat{\Theta}/a$.

5) *S.5:* Step five computes the MLE of a by substituting $\widehat{\Theta}/a$ into (27) or (28), producing $\widehat{\Theta}$ the MLE for all p parameters of the model.

The CM-steps can also be derived from the LL function in (11) by obtaining partial derivatives with respect to all the model parameters using

$$\frac{\partial \mathrm{LL}}{\partial \Theta_i} = 0 \tag{31}$$

for $(1 \le i \le p)$.

The algorithm then cycles through the p CM-steps holding the other $(p-1)$ parameters constant and then applying a numerical root finding algorithm. The cycle repeats until the convergence criterion

$$|\mathrm{LL}_j - \mathrm{LL}_{j-1}| < \varepsilon \tag{32}$$

is satisfied. This method is referred to as the LL-ECM method hereafter.

The following sections present the equations needed to apply the RLL and LL-ECM algorithms for the GO and Weibull SRGM to the models discussed in Section II to demonstrate the steps involved. LL-ECM algorithms are not presented for the context of ISS, gamma, and bathtub, as the RLL-ECM consistently exhibited better performance than the LL-ECM alternative. Nevertheless, CM-steps can be derived by following the steps described above and illustrated in the context of the GO and Weibull SRGM below.

A. Goel–Okumoto SRGM

1) LL ECM: Applying (2) to (3) for the MVF of the GO SRGM provides the instantaneous failure rate

$$\lambda(t) = abe^{-\mathrm{bt}}. \tag{33}$$

The LL function of the GO SRGM is therefore

$$\mathrm{LL}(a, b|\mathbf{T}) = -a\left(1 - e^{-\mathrm{bt}_n}\right) + \sum_{i=1}^{n} \log\left(abe^{-\mathrm{bt}_i}\right). \tag{34}$$

The CM-steps for $a^{''}$ and $b^{''}$ are obtained by differentiating (34) with respect to model parameters a and b to produce

$$a^{''} = \frac{n}{1 - e^{-b^{'}t_n}} \tag{35}$$

$$b^{''} = \frac{n}{a^{''}t_n e^{-b^{''}t_n} + \sum_{i=1}^{n} t_i}. \tag{36}$$

Equation (35) obtains the updated parameter $a^{''}$ by holding $b^{'}$ constant. However, (36) must be solved numerically with $a^{''}$ to obtain the update $b^{''}$ because a closed-form solution is not available. It is also possible to first apply (36) with $a^{'}$ and then substitute $b^{''}$ into (35). Thus, unlike the EM algorithm the CM-steps of the ECM algorithm updates just one parameter at a time.

2) Reduced LL ECM: From the LL function given in (34), the MLE of parameter a is

$$\hat{a} = \frac{n}{1 - e^{-\mathrm{bt}_n}}. \tag{37}$$

Substituting (37) into (34) produces the reduced LL function

$$\mathrm{RLL}(b|\mathbf{T}) = -n + \sum_{i=1}^{n} \log\left(\frac{nbe^{-\mathrm{bt}_i}}{1 - e^{-\mathrm{bt}_n}}\right). \tag{38}$$

Since the RLL contains only one unknown parameter, namely b, the parameter can be estimated with a single application of a numerical root finding algorithm. Thus, in cases where the RLL contains only one parameter, the ECM algorithm reduces to a simple root finding problem, requiring only a single iteration. Thus, in this case, the CM-step for parameter $b^{''}$ obtained by differentiating (38) with respect to b is identical to the traditional MLE

$$b^{''} = \frac{n}{b^{''}} - \sum_{i=1}^{n} t_i - \frac{nt_n e^{-b^{''}t_n}}{1 - e^{-b^{''}t_n}}. \tag{39}$$

B. Delayed S-Shaped SRGM

1) LL ECM: Applying (2) to (4) for the MVF of the DSS SRGM provides the instantaneous failure rate

$$\lambda(t) = ab^2te^{-\mathrm{bt}}. \tag{40}$$

The LL function of the GO SRGM is therefore

$$\mathrm{LL}(a, b|\mathbf{T}) = -a\left(1 - (1 + bt_n)e^{-\mathrm{bt}_n}\right) + \sum_{i=1}^{n} \log\left(ab^2t_ie^{-\mathrm{bt}_i}\right). \tag{41}$$

The CM-steps for $a^{''}$ and $b^{''}$ are obtained by differentiating (41) with respect to model parameters a and b to produce

$$a^{''} = \frac{n}{1-(1+b^{'}t_n)e^{-b^{'}t_n}} \tag{42}$$

$$b^{''} = -a^{''}b^{''}t_n^2 e^{-b^{''}t_n} + \frac{2n}{b^{''}} - \sum_{i=1}^{n} t_i. \tag{43}$$

Equation (42) obtains the updated parameter $a^{''}$ by holding $b^{'}$ constant. However, (43) must be solved numerically with $a^{''}$ to obtain the update $b^{''}$ because a closed-form solution is not available.

2) Reduced LL ECM: From the LL function given in (41), the MLE of parameter a is

$$\hat{a} = \frac{n}{1-(1+bt_n)e^{-\mathrm{b}t_n}}. \tag{44}$$

Substituting (44) into (41) produces the reduced LL function

$$\mathrm{RLL}(b|\mathbf{T}) = -n + \sum_{i=1}^{n} \log\left(\frac{nb^2 t_i e^{-\mathrm{b}t_i}}{1-(1+bt_n)e^{-\mathrm{b}t_n}}\right). \tag{45}$$

Similar to the GO model, since the RLL contains only one unknown parameter, namely b, the parameter can be estimated with a single application of a numerical root finding algorithm. Thus, in this case, the CM-step for parameter $b^{''}$ obtained by differentiating (45) with respect to b is identical to the traditional MLE

$$b^{''} = n\left(\frac{2}{b^{''}} - t_n\left(\frac{1-e^{-b^{''}t_n}}{1-(1+b^{''}t_n)e^{-b^{''}t_n}} + 1\right)\right) + \sum_{i=1}^{n} t_i. \tag{46}$$

C. Weibull SRGM

1) LL ECM: Applying (2) to (5) for the MVF of the Weibull SRGM provides the instantaneous failure rate

$$\lambda(t) = abct^{c-1}e^{-\mathrm{b}t^c}. \tag{47}$$

The LL function is

$$\mathrm{LL}(a,b,c|\mathbf{T}) = -a\left(1-e^{-\mathrm{b}t_n^c}\right) + \sum_{i=1}^{n}\log\left(abct_i^{c-1}e^{-\mathrm{b}t_i^c}\right). \tag{48}$$

The CM-steps for $a^{''}$, $b^{''}$, and $c^{''}$ are obtained by differentiating (48) with respect to model parameters a, b, and c

$$a^{''} = \frac{n}{1-e^{-b^{'}t_n^{c^{'}}}} \tag{49}$$

$$b^{''} = \frac{n}{a^{'}t_n^{c^{'}}e^{-b^{''}t_n^{c^{'}}} + \sum_{i=1}^{n} t_i^{c^{'}}} \tag{50}$$

and

$$c^{''} = \frac{n}{a^{'}b^{'}e^{-b^{'}t_n^{c^{''}}}\log(t_n)t_n^{c^{''}} - \sum_{i=1}^{n}\log(t_i) - b^{'}\sum_{i=1}^{n} t_i^{c^{''}}\log(t_i)}. \tag{51}$$

Parameters b and c lack closed-form solutions. Thus, numerical root finding is required to obtain $b^{''}$ and $c^{''}$.

2) Reduced LL ECM: From the LL function given in (48), the MLE of parameter a is

$$\hat{a} = \frac{n}{1-e^{-\mathrm{b}t_n^c}}. \tag{52}$$

Substituting (52) into (48) produces the reduced LL function

$$\mathrm{RLL}(b,c|\mathbf{T}) = -n + \sum_{i=1}^{n}\log\left(\frac{nbct_i^{(c-1)}e^{-\mathrm{b}t_i^c}}{1-e^{-\mathrm{b}t_n^c}}\right). \tag{53}$$

Differentiating (53) according to (29), the ECM update rules for parameters b and c are

$$b^{''} = \frac{\left(-nt_n^{c^{'}}e^{-b^{''}t_n^{c^{'}}} + \left(\frac{n}{b^{''}} - \sum_{i=1}^{n} t_i^{c^{'}}\right)\left(1-e^{-b^{''}t_n^{c^{'}}}\right)\right)}{1-e^{-b^{''}t_n^{c^{'}}}} \tag{54}$$

and

$$c^{''} = \left(\frac{-nb^{'}t_n^{c^{''}}\log(t_n)\,e^{-b^{'}t_n^{c^{''}}}}{1-e^{-b^{'}t_n^{c^{''}}}}\right) + \frac{n}{c^{''}} + \sum_{i=1}^{n}\log(t_i) - b^{'}\sum_{i=1}^{n} t_i^{c^{''}}\log(t_i). \tag{55}$$

Note that the CM expressions given in (54) and (55) can be applied in any order. Thus, it is possible to update parameter b in odd iterations and parameter c in even iterations or reverse the order of their application so that parameters c and b are updated in odd and even iterations, respectively.

D. Inflexion S-Shaped SRGM

Applying (2) to (6) for the MVF of the ISS SRGM provides the instantaneous failure rate

$$\lambda(t) = \frac{ab(1+\psi)e^{\mathrm{b}t}}{(\psi+e^{\mathrm{b}t})^2}. \tag{56}$$

Substituting (6) and (56) into (11), the LL function is

$$\mathrm{LL}(t_i;\boldsymbol{\Theta}_i) = -\frac{a(1-e^{-\mathrm{b}t_n})}{1+\psi e^{-\mathrm{b}t_n}} + \sum_{i=1}^{n}\log\left(\frac{ab(1+\psi)e^{\mathrm{b}t_i}}{(\psi+e^{\mathrm{b}t_i})^2}\right). \tag{57}$$

The MLE of parameter a is

$$\hat{a} = \frac{n(1+\psi e^{-\mathrm{b}t_n})}{1-e^{-\mathrm{b}t_n}}. \tag{58}$$

Substituting (58) into (57) produces the reduced LL function

$$\mathrm{RLL}(t_i;\Theta_i) = -n + \sum_{i=1}^{n}\log\left(\frac{n(1+\psi e^{-\mathrm{b}t_n})b(1+\psi)e^{\mathrm{b}t_i}}{(1-e^{-\mathrm{b}t_n})(\psi+e^{\mathrm{b}t_i})^2}\right). \tag{59}$$

Differentiating (59) according to (29), the IECM update rules for parameters b and ψ are

$$b'' = \sum_{i=1}^{n}\left[\left(\frac{1}{1+\psi' e^{-b''t_n}} - e^{b''t_n}\right)t_n + \left(1-\frac{2}{1+\psi' e^{-b''t_i}}\right)t_i + \frac{1}{b''} + t_i\right] \quad (60)$$

$$\psi'' = \sum_{i=1}^{n}\left[\frac{1}{\psi''+e^{b't_n}} + \frac{1}{1+\psi''} - \frac{2}{\psi''+e^{b't_i}}\right]. \quad (61)$$

Similar to the Weibull model, the initial estimates are obtained using the special case of the exponential model. Setting $\Psi^{(0)} = 0$ reduces the ISS model to the GO model. Therefore, (19) is used as an initial estimate for $b^{(0)}$.

E. Gamma SRGM

Applying (2) to (8) for the MVF of the Gamma SRGM provides the instantaneous failure rate

$$\lambda(t) = \frac{ab^c e^{-\mathrm{bt}} t^{c-1}}{\Gamma(c)}. \quad (62)$$

Substituting (8) and (62) into (11), the LL function is

$$\mathrm{LL}(t_i;\boldsymbol{\Theta}_i) = -a\int_0^{t_n}\frac{b^c x^{c-1}e^{-bx}}{\Gamma(c)}dx + \sum_{i=1}^{n}\left(\frac{ab^c e^{-\mathrm{bt}_i}t_i^{c-1}}{\Gamma(c)}\right). \quad (63)$$

The MLE of parameter a is

$$\hat{a} = \frac{n}{\left(\frac{\Gamma(c,bt_n)}{\Gamma(c)} - 1\right)} \quad (64)$$

where $\Gamma(c) = \int_0^\infty t^{c-1}e^{-c}dt$ and $\Gamma(c,bt_n) = \int_0^{bt_n} e^{-t}t^{c-1}dt$

Substituting (64) into (63) produces the reduced LL function

$$\mathrm{RLL}(t_i;\Theta_i) = -n + \sum_{i=1}^{n}\left(\frac{nb^c e^{-\mathrm{bt}_i}t_i^{c-1}}{\left(\frac{\Gamma(c,bt_n)}{\Gamma(c)} - 1\right)\Gamma(c)}\right). \quad (65)$$

Differentiating (65) according to (29), the IECM update rules for parameters b and c are

$$b'' = \frac{n}{b''}\left[c' - \frac{(b''t_n)^{c'}e^{-b''t_n}}{\Gamma(c') - \Gamma(c',b''t_n)}\right] - \sum_{i=1}^{n}t_i \quad (66)$$

$$c'' = -n\left(\log(b't_n)\boldsymbol{\Gamma} + G^{3\ 0}_{2\ 0}\left(\begin{matrix}1,1\\0,0,c''\end{matrix}\middle|b't_n\right)\right) + \digamma(c'')(1+n\boldsymbol{\Gamma}) + \sum_{i=1}^{n}\log(b't_i) \quad (67)$$

where M denotes the Meijer G-function defined as $G^{m\ n}_{p\ q}\left(\begin{matrix}a_1,\ldots,a_p\\b_1,\ldots,b_q\end{matrix}\middle|z\right)$ and $\digamma(c)$ is the digamma function, defined as $\frac{\Gamma'(c)}{\Gamma(c)}$ and

$$\boldsymbol{\Gamma} = \frac{\Gamma(c,bt_n)}{\Gamma(c) - \Gamma(c,bt_n)}$$

Similar to the Weibull and ISS models, the initial estimates are obtained using the special case of the exponential model. Setting $c^{(0)} = 0$ reduces the gamma model to the GO model. Therefore, (19) is also used as an initial estimate for $b^{(0)}$.

F. Bathtub SRGM

Applying (2) to (10) for the MVF of the bathtub SRGM provides the instantaneous failure rate

$$\lambda(t) = abct^{c-1}e^{\phi t - \mathrm{bct}^c e^{\phi t}}(c+\phi t). \quad (68)$$

Substituting (10) and (68) into (11), the LL function is

$$\mathrm{LL}(t_i;\boldsymbol{\Theta}_i) = -a(1-e^{-bce^{\phi t_n}tn^c}) + \sum_{i=1}^{n}\log\left[abct_i^{c-1}e^{\phi t_i - bct_i^c e^{\phi t_i}}(c+\phi t_i)\right]. \quad (69)$$

The MLE of parameter a is

$$\hat{a} = \frac{n}{1-e^{-bce^{\phi t_n}t_n^c}}. \quad (70)$$

Substituting (70) into (69) produces the reduced LL function

$$\mathrm{RLL}(t_i;\Theta_i) = -n + \sum_{i=1}^{n}\log\left(\frac{nbct_i^{c-1}(c+\phi t_i)e^{\phi t_i - bct_i^c e^{\phi t_i}}}{1-e^{-bct_n^c e^{\phi t_n}}}\right). \quad (71)$$

Differentiating (71) according to (29), the IECM update rules for parameters b, c and ϕ are

$$b'' = \sum_{i=1}^{n}\left[b'' - \frac{c't_n^{c'}e^{t_n^{1+c'}\phi'}}{1-e^{-b''c't_n^{c'}e^{\phi' t_n}}} + c'\left(e^{\phi' t_n}t_n^{c'} - e^{\phi' t_i}t_i^{c'}\right)\right] \quad (72)$$

$$c'' = \sum_{i=1}^{n}\left[\frac{1}{c''+\phi' t_i} + \frac{1}{c''} + \log(t_i) - \frac{b't_n^{c''}(1+c''\log(t_n))e^{\phi' t_n}}{1-e^{-b'c''t_n^{c''}e^{\phi' t_n}}} + \left(b'e^{\phi' t_n}t_n^{c''}(1+c''\log(t_n)) - b't_i^{c''}(1-c''\log(t_i))e^{\phi' t_i}\right)\right] \quad (73)$$

$$\phi'' = \sum_{i=1}^{n}\left[\frac{t_i}{(c'+\phi'' t_i)} - \frac{b'c't_n^{c+1}e^{\phi'' t_n}}{1-e^{b'c't_n^c e^{\phi'' t_n}}} + \left(b'c'\left(t_n^{c+1}e^{\phi'' t_n} - t_i^{c+1}e^{\phi'' t_i}\right) + t_i\right)\right]. \quad (74)$$

Since application of (18) lacks closed form solutions for c and ϕ, the initial estimates for b is obtained by reducing to the special case of the GO model by setting $\phi^{(0)} = 0$, which reduces to the Weibull model and subsequently setting $c^{(0)} = 1.0$ to further reduce this to the exponential, possessing the initial estimate for $b^{(0)}$ in (19).

V. Illustrations

This section illustrates the ECM algorithm through a series of examples. We first illustrate the RLL-ECM in the context

of the Weibull model and then conduct a comparative performance analysis on LL-ECM and RLL-ECM algorithms for the GO, DSS, and Weibull, suggesting that the RLL-ECM algorithm outperforms the LL-ECM algorithm. Thus, we next compare the performance of our RLL-ECM algorithm with our previously proposed EM and ECM algorithms [1] as well as the EM algorithm for the LEVMIN SRGM [2], which is equivalent to the Weibull model under parameter transformation. In some cases, the EM algorithm for the LEVMIN model outperforms our RLL-ECM algorithm. Hence, we extend the RLL-ECM algorithm to a two-stage approach that first executes the RLL-ECM for a fixed number of iterations in the first stage and then runs the Newton–Raphson method in stage two, achieving speed up that consistently outperforms the EM algorithm for the LEVMIN model, even when it is used in a two-stage approach. Finally, we assess the performance of the two-stage approach on more complex models.

A. Weibull RLL-ECM Application

This example illustrates the steps of the RLL-ECM algorithm by applying the Weibull SRGM to the SYS1 dataset [23], which consists of $n = 136$ failure times.

As noted in Section III-B, the EM and ECM algorithms lack a closed form expression for the initial value of c. However, it is possible to use the initial estimate of the exponential model given in (19) by setting $c^{(0)} = 1.0$, providing the initial value of $b^{(0)} = 0.0000404$. Since the exponential model is a special case of the Weibull, this strategy of starting from $c^{(0)} = 1.0$ performed well on all datasets considered because the EM algorithm for this simpler model often converges despite perturbations to the initial estimates [27]. The value of the likelihood function at these initial estimates is -975.899. The first iteration applies (54), holding c constant at 1.0 and solving for $b^{(1)} = 0.000034$, increasing the LL value to -974.597. Similarly, the second iteration applies (55), holding b constant at 0.000034 and solves to identify $c^{(1)} = 0.9917$, increasing the LL value to -974.172. Successive odd and even iterations update b and c, respectively. For example, iterations three and four update the parameters to $b^{(2)} = 0.0000371694$ and $c^{(2)} = 0.983453$, achieving a LL of -973.358. Thus, like the EM, the ECM algorithm improves the LL monotonically in each iteration. The iterations of the ECM algorithm continue until the error between two successive values of the LL given in (30) is less than the convergence constant $\varepsilon = 10^{-15}$. This occurs after a total of 173 iterations, including the calculation of the initial estimate of $b^{(0)}$. The resulting MLEs are $\{\hat{b} = 0.000696057, \hat{c} = 0.676739\}$ and the corresponding value of the likelihood function evaluated at these estimates is -966.080. Substituting the estimates for parameters b and c into (52) produces the MLE for the initial number of faults $\hat{a} = 172.526$.

Fig. 1 shows the final iterations of the ECM algorithm superimposed on a contour plot of the LL function. The 90° angle movements illustrate how only one parameter is updated at a time. It can also be observed that the algorithm takes smaller and smaller steps as the parameter estimates converge to the MLE. Fig. 2 shows the monotonic improvements made by the ECM in each of the 173 iterations.

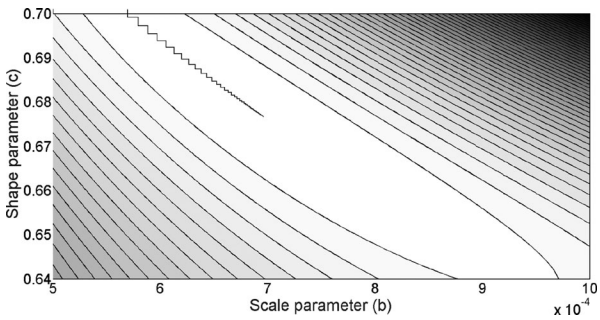

Fig. 1. Iterations of ECM superimposed on contour plot of LL function.

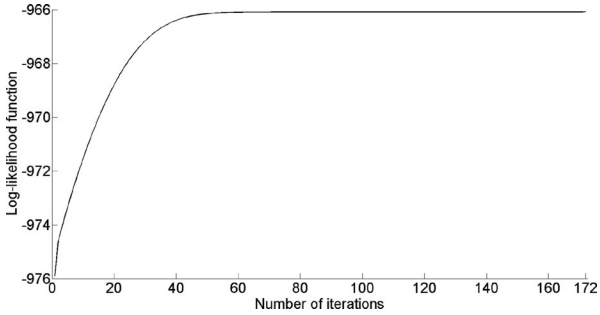

Fig. 2. Monotonic improvement of LL function in iterations of ECM.

TABLE I
COMPARISON LL AND RLL-ECM ALGORITHMS FOR GO SRGM

Dataset	LL-ECM	RLL-ECM	LL/RLL	p-value
SYS1	0.1310	0.0624	2.1000	2.82×10^{-05}
SYS2	0.2246	0.0437	5.1428	3.23×10^{-10}
SYS3	0.5647	0.1061	5.3235	2.18×10^{-10}
S2	0.0530	0.0218	2.4286	0.000208803
S27	0.0437	0.0187	2.3333	0.000238807
SS3	1.7441	0.1310	13.3095	1.09×10^{-10}
SS4	8.2119	0.0905	90.7586	3.96×10^{-08}
CSR1	0.2714	0.1810	1.5000	8.19×10^{-08}
CSR2	0.1061	0.0655	1.6191	7.93×10^{-06}
CSR3	0.1903	0.0468	4.0667	6.68×10^{-07}

B. Performance Analysis

This section compares the performance of the RLL-ECM and LL-ECM when applied to the GO and Weibull SRGM for ten failure times datasets. We then compare the performance of the RLL-ECM to our previous EM and ECM [1] algorithms as well as other previously proposed EM algorithms [2].

1) Comparison of LL-ECM and RLL-ECM Algorithms: This example compares the performance of the LL and reduced LL ECM algorithm given in Section IV. ECM algorithms for the GO and Weibull SRGM were applied to the ten failure times datasets taken from the literature [4], [23] with convergence error $\varepsilon < 10^{-10}$ in (30) and (32) for RLL and LL-ECM algorithms, respectively.

Table I reports the run times (in seconds) of the LL ECM and reduced LL ECM algorithms given in Section IV-A for the ten failure time datasets from the literature [4], [23], reporting the average time of five runs. The ECM-LL run times were obtained with the equations given in Section IV-A1, while

TABLE II
COMPARISON LL AND RLL-ECM ALGORITHMS FOR DSS SRGM

Dataset	LL-ECM	RLL-ECM	LL/RLL	p-value
SYS1	0.0905	0.0598	1.5130	0.002836542
SYS2	0.0905	0.0468	1.9333	8.29×10^{-05}
SYS3	0.2215	0.1154	1.9189	2.30×10^{-05}
S2	0.0468	0.0251	1.8750	0.00231792
S27	0.0374	0.0156	2.4000	0.00231792
SS3	0.4056	0.1498	2.7083	3.39×10^{-05}
SS4	0.4118	0.0905	4.5517	3.82×10^{-10}
CSR1	0.2527	0.2090	1.2091	1.34×10^{-05}
CSR2	0.0749	0.0655	1.1429	0.103075503
CSR3	0.0905	0.0530	1.7059	3.92×10^{-05}

TABLE III
COMPARISON LL AND RLL ECM ALGORITHMS FOR WEIBULL SRGM

Dataset	LL-ECM	RLL-ECM	LL/RLL	p-value
SYS1	1.6723	1.1669	1.4332	2.66×10^{-09}
SYS2	2.0842	0.9204	2.2644	3.49×10^{-10}
SYS3	3.4164	1.2886	2.6513	1.68×10^{-09}
S2	0.8299	0.9329	0.8896	1.90×10^{-06}
S27	0.6178	0.8611	0.7174	7.89×10^{-08}
SS3	7.4631	0.5647	13.2155	1.09×10^{-13}
SS4	6.8079	1.2262	5.5522	3.83×10^{-16}
CSR1	5.3536	5.4039	0.9908	0.076701343
CSR2	2.1934	2.6271	0.8349	3.39×10^{-05}
CSR3	0.9485	0.1934	4.9032	5.56×10^{-15}

the ECM-RLL run times were computed using the equations in Section IV-A2. The LL and RLL results are shown in the second and third column, respectively. The fourth column provides the ratio of LL and RLL run times, while the fifth column reports the p-value (Student's t-test [28]) of a two means test for equivalence in the run time of the two approaches. Because the LL/RLL ratio is greater than one Table I indicates that the run time of the ECM algorithm which uses the LL function is slower than the RLL approach for all ten datasets considered. The p-values provided in the last column of Table I indicate the RLL approach is significantly faster. Since it was noted in Section IV-A2 that the reduced LL ECM algorithm for the GO SRGM reduces to a single application of a numerical root finding algorithm for parameter b in (38), this observation indicates that Newton's method with the initial parameter estimates obtained from (18) is adequate to achieve convergence for each of the datasets considered. Thus, when the model is sufficiently simple it may be preferable to employ Newton's method with initial estimates determined by the EM algorithm.

Table II provides the run times (in seconds) of the LL and reduced LL ECM algorithms for the DSS SRGM given in Sections IV-B1 and IV-B2, respectively. The algorithms were run five times each with the initial estimates determined from (20) and the average run time computed.

In Table II, the LL/RLL ratio is greater than one for all ten datasets indicating a similar trend to Table I. The p-values also indicate that the RLL approach is significantly faster. This is because the RLL ECM algorithm for the DSS SRGM reduces to a single application of a numerical root finding algorithm for parameter b in (46). These results also suggests that when the model is sufficiently simple Newton's method with initial estimates determined by the EM algorithm may be preferred.

Table III lists the run times (in seconds) of the LL and reduced LL ECM algorithms for the Weibull SRGM in Section IV-C. The ECM-LL run times were obtained with the equations given in Section IV-C1, while the ECM-RLL run times were computed using the equations in Section IV-C2. These algorithms were run five times each with the initial estimates determined from (19) by setting $c^{(0)} = 1.0$ and the average run time computed.

Table III indicates that the reduced LL ECM algorithm significantly outperformed the log-likelihood ECM algorithm on six out of ten datasets where the ratio LL/RLL was greater than 1.0.

TABLE IV
COMPARISON OF EM [1], ECM [1], AND RLL-ECM ALGORITHM

Factor (ρ)	EM [1]	ECM [1]	RLL-ECM	$\frac{\text{EM [1]}}{\text{RLL-ECM}}$	$\frac{\text{ECM [1]}}{\text{RLL-ECM}}$
0.25	2.695	2.310	0.012	230.342	197.434
0.50	2.232	1.994	0.010	220.116	196.645
0.75	1.808	1.840	0.005	386.322	393.151
0.90	1.764	1.689	0.005	323.075	309.338
1.25	1.826	1.780	0.006	292.626	285.255
1.50	1.888	1.761	0.005	345.785	322.525
1.75	1.965	1.966	0.006	314.902	315.062
2.00	2.123	1.983	0.006	340.222	317.786

However, the LL ECM algorithm outperformed the reduced LL ECM algorithm on the S2, S27, CSR1, and CSR2 datasets. However, it is important to note that the reduced LL ECM algorithm outperformed the LL ECM algorithm by as much as 13.21 times, but that the reduced LL ECM algorithm never required more than 140% of the time taken by the LL ECM algorithm, since $1/0.7174 = 1.39$. These observations suggest that the relative attractiveness of the reduced LL ECM algorithm may increase as model complexity increases because it can simplify the computation by reducing the number of model parameters to $(p-1)$. We note that LL and RLL ECM algorithms as well as all methods considered in the examples that follow produce the same parameter estimates. However, we do not report these details because the primary focus of this paper is to determine which method obtains the MLEs most efficiently.

2) Comparison of EM, ECM, and RLL-ECM Algorithms: This example applies the EM and ECM algorithms introduced in [1] as well as the RLL-ECM algorithm to fit the Weibull SRGM to a simulated dataset [1] with parameters $(a = 200, b = 4.0, c = 2.0)$.

Table IV lists the run time (in seconds) required to achieve a convergence error of less than $\varepsilon = 10^{-10}$ as specified in (30) and (32), respectively. For the sake of argument, several initial estimates that were a multiplicative factor ρ of the true MLEs ($\Theta_0 = \rho \times \hat{\Theta}$) were tested because this was the method applied in [1] to demonstrate convergence of the EM and ECM algorithms proposed there. Here, $\rho < 1$ indicates that the initial parameter estimates were smaller than the MLEs, while $\rho > 1$ indicates that the initial estimates were larger. These initial estimates were used as input to the EM and ECM algorithms of [1]

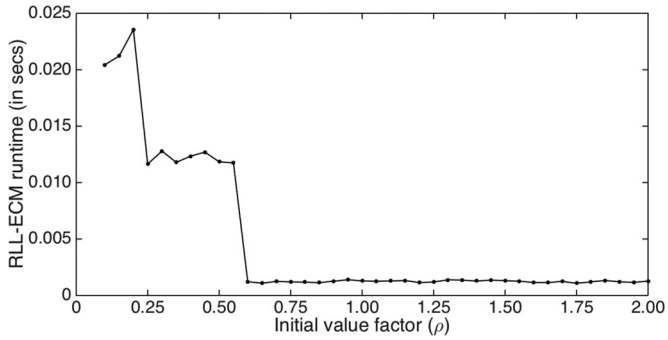

Fig. 3. Run time of RLL-ECM for range of initial parameter estimates.

TABLE V
COMPARISON OF WEIBULL RLL-ECM AND LEVMIN EM [2] RUN TIMES

Datasets	EM [2]	RLL-ECM	$\frac{\text{EM [2]}}{\text{RLL-ECM}}$	p-value
SYS1	2.0062	1.1669	1.7192	2.03×10^{-09}
SYS2	3.0015	0.9204	3.2610	3.41×10^{-11}
SYS3	5.9686	1.2886	4.6321	2.43×10^{-08}
S2	0.7301	0.9329	0.7826	3.16×10^{-08}
S27	0.2402	0.8611	0.2790	3.24×10^{-10}
SS3	34.3733	0.5647	60.8674	1.62×10^{-08}
SS4	14.6547	1.2262	11.9516	2.51×10^{-10}
CSR1	0.8736	5.4039	0.1617	4.89×10^{-11}
CSR2	0.5242	2.6271	0.1995	3.96×10^{-15}
CSR3	3.7783	0.1934	19.5323	2.68×10^{-13}

as well as our RLL-ECM to ensure comparability of the results despite the fact that it is impossible to know the MLEs in advance to apply such scaling by a multiplicative factor. For each value of ρ, the RLL-ECM algorithm was run 20 times for each initial estimate and the average computed.

The last two columns of Table IV respectively show the ratio of the performance between the EM [1] and RLL-ECM as well as the ECM algorithm [1] and RLL-ECM. The ratio of the performance between the EM and RLL-ECM is 220.116 and 386.322, indicating that the RLL-ECM outperforms the EM algorithm by at least two orders of magnitude on this dataset. Similarly, Table IV indicates that the ratio of the performance of the ECM and RLL-ECM ranges between 196.645 and 393.151, indicating that the RLL-ECM algorithm is approximately 200 to 400 times faster than the ECM algorithm [1] which utilizes numerical integration.

Fig. 3 shows the run time of the RLL-ECM in seconds for a range of initial parameter estimates with a multiplicative factor of ρ between 0.05 and 2.0. Fig. 3 indicates that the RLL-ECM run time is larger when $\rho < 1$, but that the run time never exceeds 0.025 s. As might be expected, values of $\rho \approx 1$ exhibit the best performance and the run time improves as ρ approaches 1.0. The range from 0.6 to 2.0 exhibits little variation in the run time, indicating that if the initial estimates are not too far from the MLEs, the algorithm exhibits better performance, converging faster. Furthermore, the run time for $\rho \geq 0.6$ never exceeds one-hundredth of a second.

While the previous experiments enable objective comparison of the EM, ECM [1], and RLL-ECM algorithms, it is not realistic to assume that the MLEs are already available. Otherwise, there would be no need for such algorithms. In practice, it will be necessary to select initial estimates. For example, using the initial estimate from (19) on the simulated dataset, the RLL-ECM algorithm required 0.183684 s to fit the Weibull SRGM, whereas the EM and ECM required 217.258 and 299.131 s, respectively. Thus, the RLL-ECM also outperformed the EM and ECM by factors of 1182.8 and 1628.5, respectively, when initial parameter estimates based on the EM algorithm are used.

3) Comparison of EM [2] Algorithm for LEVMIN Model and RLL-ECM Algorithms: The EM algorithm for the LEVMIN model [2] is equivalent to the Weibull model under parameter transformation and therefore also suitable for comparative performance analysis. For equitable comparison, we implemented the update rules for the EM algorithm of the LEVMIN model [2] and RLL-ECM algorithm for Weibull model in Mathematica and ran them until the convergence criterion $\varepsilon < 10^{-10}$ specified in (30) was achieved. We ran the algorithms on the ten failure time datasets considered in Section V-B1 for five times on each dataset and the average computed. Initial parameters estimates for the RLL-ECM were chosen according to (19) by setting $c^{(0)} = 1.0$, while initial parameter estimates for the LEVMIN model were chosen according to the equations given in [2].

Table V reports the averages of five runs of the Weibull RLL-ECM algorithm and EM algorithm for the LEVMIN model and the ratio of the p-value for equality in the mean run times of the algorithms. All times reported are in seconds.

Row one of Table V indicates that for the SYS1 dataset, the RLL-ECM algorithm for the Weibull model took on average 1.1669 s, while the average run time of the EM algorithm for the LEVMIN model was 2.0062 s. Thus, the averages of the five runs suggest that the RLL-ECM algorithm is 1.7192 times faster than the EM algorithm for fitting the LEVMIN model. The p-value of a two means test was 2.03×10^{-09}, which is extremely significant. Moreover, there was little variation in the individual runs, with standard deviations of 0.0131 and 0.0283 for the RLL-ECM and EM algorithms, respectively. The ratio of EM [2]/RLL-ECM is greater than one for six out of ten datasets, indicating that the RLL-ECM outperformed the EM algorithm six times and the p-value is extremely significant in all the six cases. Moreover, the RLL-ECM outperformed the EM by as much as 60.8674 times, but the EM was never more than 6.18 times faster $(1/0.1617)$ than the RLL-ECM and in all ten cases the run time of the RLL-ECM was never more than 5.4 s.

C. Two-Stage Algorithm

Application of RLL-ECM algorithm to the SYS1 dataset for the ISS, gamma, and bathtub models with a convergence error of $\varepsilon < 10^{-10}$ required run time of 29.55, 877.22, and 88.89 s, respectively. To enhance the run time of the ECM-RLL approach while preserving stability, this example seeks to improve the performance of maximum-likelihood estimation by combining the stability of the RLL-ECM and the performance of Newton's method. This two-stage approach improves the performance of

TABLE VI
RUN TIMES OF RLL-ECM AND TWO-STAGE ALGORITHMS FOR WEIBULL SRGM

Datasets	RLL-ECM	Stage I	Stage II	Two-stage total	$\frac{\text{RLL-ECM}}{\text{RLL-ECM Two-Stage}}$	p-value
SYS1	1.1669	0.0374	0.0156	0.0530	22.00	7.44×10^{-14}
SYS2	0.9204	0.0343	0.0156	0.0499	19.67	4.89×10^{-65}
SYS3	1.2886	0.0499	0.0218	0.0718	18.77	1.46×10^{-13}
S2	0.9329	0.0312	0.0125	0.0468	19.93	5.65×10^{-09}
S27	0.8611	0.0156	0.0062	0.0218	39.43	3.13×10^{-10}
SS3	0.5647	0.0499	0.0187	0.0686	8.62	2.08×10^{-14}
SS4	1.2262	0.0406	0.0156	0.0562	21.83	7.12×10^{-10}
CSR1	5.4039	0.0780	0.0343	0.1123	50.94	1.19×10^{-10}
CSR2	2.6271	0.0406	0.0156	0.0562	52.63	4.40×10^{-19}
CSR3	0.1934	0.0218	0.0125	0.0343	7.05	6.90×10^{-10}

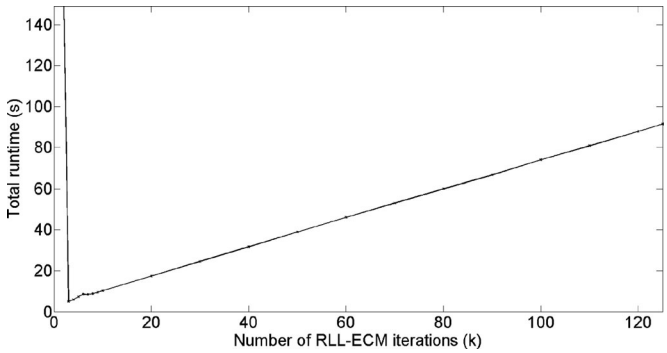

Fig. 4. Run time of two-stage algorithm for bathtub model.

complicated models and further enhances the performance of the Weibull ECM-RLL.

Fig. 4 shows the run time of the two-stage algorithm for the bathtub model when applied to the SYS1 dataset with a convergence error of less than 10^{-10}. The *x*-axis shows the number of iterations k of the RLL-ECM algorithm employed to obtain initial estimates for Newton's method and the *y*-axis shows the total run time of the two algorithms combined. Fig. 4 shows an infinite run time for $k = 0, 1$, and 2, indicating nonconvergence of the algorithms. It is important to note that $k = 0$ indicates that Newton's method alone is not sufficient to estimate the model parameters with the default initial estimates. The algorithm converges when the RLL-ECM is run for a minimum of three iterations for this model. The minimum number of iterations for the other models and datasets may vary.

The increasing trend for $k \geq 3$ in Fig. 4 indicates that the run time increases as the number of times the RLL-ECM algorithm is run increases, suggesting that switching to Newton's method sooner converges faster. Therefore, a sufficient number of RLL-ECM iterations combined with Newton's method can improve stability while preserving performance. For example, running only the RLL-ECM achieves convergence in 125 iterations but requires 88.889 s, while the hybrid approach requires only 5.070 s when $k = 3$, which makes the hybrid approach 17.53 times faster than running the RLL-ECM algorithm without switching to Newton's method. Thus, considering the hybrid approach can improve the performance of a complex SRGMs significantly. Nevertheless, it is important to recognize that selecting the optimal value of k cannot be performed *a priori*. However, a simple strategy is to run the RLL-ECM for a specified number of iterations and then switch to Newton's Method. If this fails to converge, continue to run the RLL-ECM for $2k$ iterations and try Newton's Method again, alternating between the RLL-ECM and Newton's Method until convergence is achieved. Hence, while it is possible that the two-stage algorithm may diverge if the initial input to Newton's Method is too far from the MLE, this can be avoided in practice essentially guaranteeing convergence.

Table VI compares the run time of RLL-ECM algorithm and two-stage algorithms for the Weibull SRGM on the ten failure times datasets. These run times reported are an average of five runs. In each run, on a dataset five iterations of the RLL-ECM algorithm were performed in Stage I and Newton's method in Stage II. All run times are given in seconds. The sixth column of Table VI illustrates that the ratio of average RLL-ECM and two-stage algorithm run times are always greater than one, suggesting that the two-stage algorithm consistently outperforms the RLL-ECM algorithm. This ratio indicates that the speed up attained by the two-stage algorithm ranges between 7 and 52. In all cases, the p-value for a two means test between the run times of the RLL-ECM and two-stage algorithm is extremely small, suggesting that the speedup achieved by the two-stage algorithm is highly statistically significant.

Table VII compares the run time of two-stage algorithms for the Weibull and LEVMIN [2] models on the ten datasets. The sixth column indicates the ratio of two-stage algorithms utilizing the EM [2] and RLL-ECM algorithms in Stage I. In contrast to, the results reported in Table V of Example V-B3, the ratios in the sixth column of Table VII indicate that the two-stage algorithm that utilizes the RLL-ECM algorithm in Stage I outperforms the two-stage algorithm that utilizes the EM [2] algorithm in Stage I for every single dataset, despite the improved performance in the EM [2] algorithm when incorporated into a two-stage algorithm as can be seen by comparing the results noted above in column five of the Table VII.

Tables VIII–X report the average run time of two-stage algorithms for the ISS, gamma, and bathtub models. Stages I and II in Tables VIII–X correspond to the times take for the RLL-ECM and Newton's method, respectively. Row one of

TABLE VII
RUN TIMES OF TWO-STAGE ALGORITHMS FOR WEIBULL AND LEVMIN [2] SRGM

Datasets	Stage I	Stage II	LEVMIN two-stage total	RLL-ECM two-stage total	$\frac{\text{LEVMIN Two-Stage Total}}{\text{RLL-ECM Two-Stage Total}}$	p-value
SYS1	0.0218	0.1872	0.2090	0.0530	3.94	1.32×10^{-09}
SYS2	0.0125	0.1310	0.1435	0.0499	3.07	2.21×10^{-04}
SYS3	0.0281	0.2964	0.3245	0.0718	4.72	5.89×10^{-08}
S2	0.0125	0.0718	0.0842	0.0468	1.81	3.04×10^{-04}
S27	0	0.0593	0.0593	0.0218	2.71	5.25×10^{-05}
SS3	0.0312	0.3869	0.4181	0.0686	6.38	1.09×10^{-09}
SS4	0.0251	0.2808	0.3058	0.0562	5.44	2.67×10^{-11}
CSR1	0.0530	0.6053	0.6583	0.1123	6.21	1.86×10^{-08}
CSR2	0.0187	0.2402	0.2590	0.0562	5.19	5.86×10^{-08}
CSR3	0.0156	0.1248	0.1404	0.0343	5.11	1.54×10^{-06}

TABLE VIII
RUN TIMES OF TWO-STAGE ALGORITHMS ISS SRGM

Datasets	Stage I	Stage II	ISS
SYS1	0.5429	37.2187	37.7616
SYS2	0.3463	6.0903	6.4366
SYS3	0.8424	0.3713	1.2137
S2	0.2184	6.1558	6.3742
S27	0.1654	0.5086	0.6739
SS3	1.1451	0.4992	1.6443
SS4	0.7862	0.3307	1.1171
CSR1	4.4772	30.6542	35.1314
CSR2	0.5179	24.5296	25.0475
CSR3	0.4150	0.6022	1.0171

TABLE IX
RUN TIMES OF TWO-STAGE ALGORITHMS FOR GAMMA SRGMS

Datasets	Stage I	Stage II	Gamma
SYS1	2.0093	0.5928	2.6021
SYS2	1.5351	0.5897	2.1247
SYS3	1.9999	0.5990	2.5990
S2	1.9157	0.7987	2.7144
S27	2.0467	4.7456	6.7923
SS3	1.5226	0.2995	1.8221
SS4	1.7411	0.44304	2.1840
CSR1	1.9157	4.5864	6.5021
CSR2	1.9126	0.3838	2.2963
CSR3	2.9921	0.7238	3.7159

TABLE X
RUN TIMES OF TWO-STAGE ALGORITHMS FOR BATHTUB SRGMS

Datasets	Stage I	Stage II	Bathtub
SYS1	1.3416	4.1964	5.5380
SYS2	0.8455	0.9329	1.7784
SYS3	2.1279	2.4523	4.5802
S2	0.5117	0.8798	1.3915
S27	0.3588	0.4493	0.8081
SS3	2.6208	4.1278	6.7486
SS4	1.8315	54.2135	56.0450
CSR1	4.1216	3.4383	7.5598
CSR2	1.3416	1.1170	2.4586
CSR3	0.9329	1.2231	2.15594

Table VIII shows the run time for the SYS1 dataset, where the two-stage algorithm for the ISS model exhibited a run time of 37.7616 s which is slower than the 29.55 s required of the ECM-RLL alone. However, this can be improved by running the RLL-ECM algorithm ten times in the first stage followed by Newton's method. This produces a total run time of 2.4929 s, requiring 1.08577 and 1.40713 s in Stages I and II, respectively. Increasing the number of RLL-ECM iterations can also improve the performance on the CSR1 and CSR2 datasets and suggests that parameter values determined by the ECM-RLL that are sufficiently close to the MLE improve the performance in stage II.

Tables IX and X indicate that the run times of the two-stage algorithm on the gamma and bathtub models for the SYS1 dataset are 2.6021 and 5.5380 s, respectively, compared to 877.22 and 140.89 s for the ECM-RLL algorithm alone. Thus, the two-stage algorithm is 337.12 and 25.44 times faster than RLL-ECM alone for gamma and baththub models, respectively. Moreover, executing ten iterations of the ECM-RLL algorithm in stage one on the SS4 dataset reduces the total run time from 56.0450 to 6.4137 s with 3.7409 and 2.67282 s spent in stage I and stage II, respectively.

VI. CONCLUSION AND FUTURE RESEARCH

This paper presents an ECM algorithm to estimate the parameters of a nonhomogeneous Poisson process software reliability growth model. Two variants were considered, including one based on the full LL expression referred to as the LL-ECM algorithm and a second based on a reduced LL expression referred to as the RLL-ECM algorithm. The RLL-ECM algorithm exhibited better performance on simpler models. Thus, the RLL-ECM algorithm was compared to previous EM [2] and ECM [1] algorithms. The results suggested that our approach achieved statistically significant improvements in performance over alternative approaches. Specifically, our ECM algorithms achieved two orders of magnitude speed up over the EM and ECM algorithms of [1] when their experimental methodology was considered and three orders of magnitude when knowledge of the MLEs is removed, whereas our approach outperforms was as much as 60 times faster than the EM algorithms of [2]. We subsequently proposed a two-stage algorithm that combined the EM and ECM algorithms with Newton's method to improve

performance while preserving stability, where it was observed that our two-stage ECM algorithm consistently outperformed a two-stage EM algorithm based on [2].

Future research will seek better strategies to identify initial estimates for parameters that lack a closed form solution. We will also compare the performance of the two-stage algorithm proposed here with alternative combinations of algorithms.

REFERENCES

[1] P. Zeephongsekul, C. Jayasinghe, L. Fiondella, and V. Nagaraju, "Maximum likelihood estimation of parameters of NHPP software reliability models using EM and ECM algorithms," *IEEE Trans. Rel.*, vol. 65, no. 3, pp. 1571–1583, Sep. 2016.

[2] H. Okamura and T. Dohi, "Application of EM algorithm to NHPP-based software reliability assessment with ungrouped failure time data," in *Proc. Stoch. Rel., Maintenance Model.*, 2013, pp. 285–313.

[3] L. Leemis, *Reliability: Probabilistic Models and Statistical Methods*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.

[4] M. Lyu, Ed., *Handbook of Software Reliability Engineering*. New York, NY, USA: McGraw-Hill, 1996.

[5] S. Yamada, *Software Reliability Modeling: Fundamentals and Applications*. Berlin, Germany: Springer, 2014.

[6] S. Ross, *Introduction to Probability Models*, 8th ed. New York, NY, USA: Academic, 2003.

[7] M. Zhao and M. Xie, "On maximum likelihood estimation for a general non-homogeneous Poisson process," *Scand. J. Statist.*, vol. 23, no. 4, pp. 597–607, Dec. 1996.

[8] S. Yamada and Y. Tamura, *OSS Reliability Measurement and Assessment*. Berlin, Germany: Springer, 2016.

[9] P. Kapur, H. Pham, A. Gupta, and P. Jha, *Software Reliability Assessment With OR Applications*. Berlin, Germany: Springer, 2011.

[10] S. Yamada, H. Ohtera, and H. Narihisa, "Software reliability growth models with testing-effort," *IEEE Trans. Rel.*, vol. R-35, no. 1, pp. 19–23, Apr. 1986.

[11] K. Okumoto and A. Goel, "Optimum release time for software systems based on reliability and cost criteria," *J. Syst. Softw.*, vol. 1, pp. 315–318, 1980.

[12] S. Hossain and R. Dahiya, "Estimating the parameters of a non-homogeneous Poisson-process model for software reliability," *IEEE Trans. Rel.*, vol. 42, no. 4, pp. 605–612, Dec. 1993.

[13] E. Elsayed, *Reliability Engineering*, 2nd ed. Hoboken, NJ, USA: Wiley, 2012.

[14] R. Burden and J. Faires, *Numerical Analysis*, 8th ed. Belmont, CA, USA: Brooks/Cole, 2004.

[15] A. Dempster, N. Laird, and D. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. Roy. Statist. Soc., Series B*, vol. 39, no. 1, pp. 1–38, Jan. 1977.

[16] X. Meng and D. Rubin, "Maximum likelihood estimation via the ECM algorithm: A general framework," *Biometrika*, vol. 80, no. 2, pp. 267–278, 1993.

[17] A. Goel and K. Okumoto, "Time-dependent error-detection rate model for software reliability and other performance measures," *IEEE Trans. Rel.*, vol. R–28, no. 3, pp. 206–211, Aug. 1979.

[18] S. Yamada, M. Ohba, and S. Osaki, "S-shaped reliability growth modeling for software error detection," *IEEE Trans. Rel.*, vol. R–32, no. 5, pp. 475–484, Dec. 1983.

[19] S. Yamada and S. Osaki, "Reliability growth models for hardware and software systems based on nonhomogeneous Poisson process: A survey," *Microelectron. Rel.*, vol. 23, no. 1, pp. 91–112, Dec. 1983.

[20] M. Ohba, "Inflection S-shaped software reliability growth model," in *Stochastic Models in Reliability Theory*. Berlin, Germany: Springer, 1984, pp. 144–162.

[21] M. Zhao and M. Xie, "On maximum likelihood estimation for a general non-homogeneous poisson process," *Scan. J. Statist.*, vol. 23, pp. 597–607, 1996.

[22] L. Fiondella and S. Gokhale, "Software reliability model with bathtub-shaped fault detection rate," in *Proc. Annu. Rel. Maintainability Symp.*, Orlando, FL, USA, Jan. 2011, pp. 1–6.

[23] L. Fiondella, "Software Failure and Reliability Assessment Tool (SFRAT)," 2016. [Online]. Available: http://sasdlc.org/lab/projects/srt.html

[24] H. Pham, *Software Reliability*. New York, NY, USA: Springer-Verlag, 2000.

[25] L. Lee, "Testing adequacy of the Weibull and log linear rate models for a Poisson process," *Technometrics*, vol. 22, pp. 195–199, 1980.

[26] J. Horgan and S. London, "ATAC: A data flow coverage testing tool for C," in *Proc. Symp. Assessment Quality Softw. Develop. Tools*, New Orleans, LA, USA, May 1992, pp. 2–10.

[27] H. Okamura, Y. Watanabe, and T. Dohi, "An iterative scheme for maximum likelihood estimation in software reliability modeling," in *Proc. 14th Int. Symp. Softw. Rel. Eng.*, Nov. 2003, pp. 246–256.

[28] D. Wackerly, W. Mendenhall, and R. Scheaffer, *Mathematical Statistics With Applications*. 5th ed. Belmont, CA, USA: Duxbury, Cengage Learning, 1996.

Vidhyashree Nagaraju (M'15) received the B.E. degree in electronics and communication engineering from Visvesvaraya Technological University, Belagavi, India, in 2011. She received the M.S. degree in electrical and computer engineering from the University of Massachusetts Dartmouth, North Dartmouth, MA, USA, in 2015, where she is currently working toward the Ph.D. degree in the Department of Electrical and Computer Engineering.

Lance Fiondella (M'05) received the Ph.D. degree in computer science and engineering from the University of Connecticut, Storrs, CT, USA, in 2012.

He is currently an Assistant Professor in the Department of Electrical and Computer Engineering, University of Massachusetts Dartmouth, North Dartmouth, MA, USA.

Dr. Fiondella served as the Vice-Chair of IEEE Standard 1633, IEEE Recommended Practice on Software Reliability (2013–2015), and is an elected member of the Administrative Committee of the IEEE Reliability Society (2015–2017). He received the 2007 scholarship from the IEEE Reliability Society and several conference paper awards.

Panlop Zeephongsekul (M'14) received the B.Sc. (Hons.) degree in statistics from Melbourne University, Melbourne, Australia, and the Ph.D. degree in mathematical statistics from the University of Western Australia, Crawley, Australia.

He is a Professor in the School of Mathematical and Geospatial Statistics, Royal Melbourne Institute of Technology, Melbourne, Australia. His research interests include stochastic point processes, fuzzy sets, game theory, supply chain, queuing theory, and software reliability. He has published extensively in all these areas. His main teaching areas are applied statistics, statistical inference, linear models, and design of experiments.

Chathuri L. Jayasinghe received the B.Sc. (Hons.) degree in statistics from the University of Sri Jayewardenepura, Nugegoda, Sri Lanka, in 2007, the M.Sc. degree (with high distinction) in statistics and operations research in 2009 and the Ph.D. degree in statistics in 2013, both from the Royal Melbourne Institute of Technology (RMIT), Melbourne, Australia.

She was a Researcher in the School of Mathematical and Geospatial Sciences, RMIT University and she is currently working with the University of Sri Jayewardenepura as a Senior Lecturer in statistics. Her research interests include software reliability growth models, nonparametric and parametric estimation methods, and survival analysis. Her work has been published in journals such as IEEE TRANSACTIONS ON RELIABILITY, the *Journal of Statistical Planning and Inference*, the *Communications in Statistics (Theory and Methods)*, and the *International Journal of Reliability, Quality and Safety Engineering*.

Thierry Wandji (M'04) received the Ph.D. degree in systems engineering from George Washington University, Washington, DC, USA, in 2015.

He is currently working as the electromagnetic interference lead engineer for multiplatform avionics at the Naval Air Systems Command (NAVAIR), Patuxent River, MD, USA. He provides Electromagnetic Environmental Effects engineering and spectrum acquisition support in concert with program requirements, study, review, and evaluates new or modified aircraft, complex avionics systems, communications equipment, subsystems, and makes assessments regarding the electromagnetic integrity of these systems. Prior to joining NAVAIR, he was a Radar System Engineer at Rockwell Collins. His research interest include Software Reliability Modeling.

Dr. Wandji is a member of the National Society of Black Engineers, the International Council on Systems Engineering, and the Electromagnetic Compatibility Society.