SOFTWARE RELIABILITY MODELLING AND ESTIMATION TECHNIQUES

Syracuse University

Amrit L. Goel
This report presents the results of the software reliability modelling and estimation research pursued under Contract F30602-78-C-0351 during the period October 1978 - October 1981. Two new models of very general applicability are introduced and the necessary mathematical and practical details are developed in this report. A new methodology for determining when to stop testing and start using software is described and developed. (Cont'd on reverse)
Finally, a new model for analyzing the operational performance of a combined hardware-software system is reported even though it was not a part of the original research plan.
ACKNOWLEDGEMENT

Much of the work reported here was pursued with K. Okumoto, who was a research assistant at Syracuse University, and is currently employed by Bell Laboratories. I owe him special thanks for his diligence in the pursuit of this work. Hardware-software reliability modelling research was done with J. Soenjoto, who is currently employed by the Indonesian Government. His help with this work is greatly appreciated.

Messers Ashish Deb and Peter Valdes provided invaluable assistance with data analyses and programming, and I thank them for their work. I have also benefited greatly from discussions with Peter Valdes regarding the ideas about software reliability described in Chapter 1.

Finally, I would like to thank Messers Alan Sukert and Joseph Cavano of RADC for their many suggestions regarding the direction of this research work.
# Table of Contents

## List of Tables

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
</table>

## List of Figures

1. **Introduction and Overview** ........................................... 1-1
   1.1 Software Errors and Their Sources ................................ 1-4
   1.2 Software Error Classification ...................................... 1-9
     1.2.1 Syntax Errors .................................................. 1-9
     1.2.2 Semantic Errors ............................................... 1-9
     1.2.3 Runtime Errors ................................................. 1-10
     1.2.4 Specification Errors .......................................... 1-13
     1.2.5 Performance Errors ............................................ 1-13
   1.3 Software Reliability ................................................ 1-15
   1.4 Approaches for Enhancing Software Reliability .................. 1-21
   1.5 Software Reliability Models ...................................... 1-24
     1.5.1 Failure Rate Based Models .................................... 1-26
     1.5.2 Combinatorial or Error-Seeding Models ........................ 1-36
     1.5.3 Input Domain Based Models .................................... 1-37

2. **A Time Dependent Fault Detection Rate Model** ...................... 2-1
   2.1 Introduction ....................................................... 2-1
   2.2 Model Development ................................................ 2-4
     2.2.1 Deterministic Analysis of Software Failure Process ...... 2-4
     2.2.2 Stochastic Analysis of Software Failure Process .......... 2-7
   2.3 Software Performance Measures .................................... 2-12
     2.3.1 Number of Software Faults Detected by t .................. 2-12
     2.3.2 Number of Remaining Faults .................................. 2-13
     2.3.3 Conditional Distribution and Expectation of $\hat{N}(t)$ .... 2-14
   2.4 Software Reliability and Distribution of Time Between Failures .. 2-18
     2.4.1 Software Reliability .......................................... 2-18
     2.4.2 Conditional Distribution of $X_k|s_{k-1}$ ................. 2-22
     2.4.3 Joint Density of Waiting Times ............................... 2-23
     2.4.4 Joint Counting Probability ................................... 2-24
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.5</td>
<td>Estimation of Model Parameters from Failure Data</td>
<td>2-25</td>
</tr>
<tr>
<td>2.5.1</td>
<td>Estimation When Cumulative Failures are Given.</td>
<td>2-26</td>
</tr>
<tr>
<td>2.5.1.1</td>
<td>Confidence Region for (a,b).</td>
<td>2-30</td>
</tr>
<tr>
<td>2.5.2</td>
<td>Estimation When Times Between Failures are Given.</td>
<td>2-34</td>
</tr>
<tr>
<td>2.6</td>
<td>Goodness-of-Fit Test</td>
<td>2-38</td>
</tr>
<tr>
<td>2.7</td>
<td>Analysis of Failure Data from Naval Tactical Data System (NTDS).</td>
<td>2-43</td>
</tr>
<tr>
<td>2.8</td>
<td>Analysis of Failure Data from a Large Scale Software System</td>
<td>2-56</td>
</tr>
<tr>
<td>2.8.1</td>
<td>Failure Data</td>
<td>2-56</td>
</tr>
<tr>
<td>2.8.2</td>
<td>Estimation of Parameters</td>
<td>2-59</td>
</tr>
<tr>
<td>2.8.3</td>
<td>Goodness-of-Fit Test</td>
<td>2-62</td>
</tr>
<tr>
<td>2.8.4</td>
<td>Confidence Regions for (a,b)</td>
<td>2-68</td>
</tr>
<tr>
<td>2.8.5</td>
<td>Variance-Covariance Matrix for (â,b̂)</td>
<td>2-69</td>
</tr>
<tr>
<td>2.8.6</td>
<td>Number of Remaining Errors</td>
<td>2-71</td>
</tr>
<tr>
<td>2.8.7</td>
<td>Software Reliability</td>
<td>2-73</td>
</tr>
<tr>
<td>2.9</td>
<td>Analysis of Failure Data from Command and Control Systems</td>
<td>2-78</td>
</tr>
<tr>
<td>2.10</td>
<td>Analyses of Various Types of Errors from a Real-Time Control System</td>
<td>2-90</td>
</tr>
<tr>
<td>2.10.1</td>
<td>Error Data</td>
<td>2-91</td>
</tr>
<tr>
<td>2.11</td>
<td>Analysis of Failure Data from the Apollo Project</td>
<td>2-101</td>
</tr>
<tr>
<td>2.12</td>
<td>Analysis of Data from a Large Avionics Real-Time System</td>
<td>2-109</td>
</tr>
<tr>
<td>3.0</td>
<td>SOFTWARE FAULT OCCURRENCE PROCESS WITH INCREASING/DECREASING ERROR DETECTION RATE</td>
<td>3-1</td>
</tr>
<tr>
<td>3.1</td>
<td>Introduction</td>
<td>3-1</td>
</tr>
<tr>
<td>3.2</td>
<td>Model Development</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.1</td>
<td>Assumptions</td>
<td>3-3</td>
</tr>
<tr>
<td>3.2.2</td>
<td>Expression for m(t)</td>
<td>3-5</td>
</tr>
<tr>
<td>3.2.3</td>
<td>Fault Detection Rate</td>
<td>3-8</td>
</tr>
<tr>
<td>3.2.4</td>
<td>Failure Counting Process</td>
<td>3-11</td>
</tr>
<tr>
<td>3.3</td>
<td>Software Effectiveness Measures</td>
<td>3-13</td>
</tr>
<tr>
<td>3.3.1</td>
<td>Distribution of the Number of Faults Detected or Failures Observed</td>
<td>3-13</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Number of Faults Remaining in the System</td>
<td>3-14</td>
</tr>
</tbody>
</table>
3.3.3 Conditional Distribution of \( \bar{N}(t) \) 3-15
3.3.4 Joint Counting Probability 3-16
3.4 Software Reliability and Distribution of Time Between Software Failures 3-18
3.5 Estimation of Model Parameters from Failure Data 3-21
3.5.1 Maximum Likelihood Estimation When Data on Cumulative Software Failures are Given 3-21
3.5.1.1 Variance-Covariance of \( \hat{a}, \hat{b}, \text{and} \hat{c} \) 3-26
3.5.2 Maximum Likelihood Estimation of Parameters When Data on Times Between Software Failures are Given 3-29
3.6 Analysis of Failure Data from a Large Scale Software System 3-32
3.6.1 Estimation of Parameters 3-32
3.6.2 Confidence Bounds 3-39
3.7 Analysis of Failure Data from Naval Tactical Data System (NTDS) 3-45

4. OPTIMUM SOFTWARE RELEASE TIME 4-1
4.1 Introduction 4-1
4.2 Software Release Time Based on Reliability Criterion 4-3
4.3 Optimum Release Time Based on Cost Criterion (Model of Section 2) 4-7
4.3.1 Cost Model and Optimal Policy 4-7
4.3.2 Sensitivity Analysis of \( T^* \) 4-12
4.4 Optimum Release Time Based on Cost Criterion (Model of Section 3) 4-15
4.4.1 Cost Model and Optimal Policy 4-15

5. BIBLIOGRAPHY 5-1

APPENDIX

A1. Introduction A-1
A2.1 System Description and Model Developments A-3
A2.2 Distribution of Time to a Specified Number of Remaining Errors in a Software System A-17
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>A2.2.1</td>
<td>Distribution of $T_{N,n}$</td>
<td>A-17</td>
</tr>
<tr>
<td>A2.2.2</td>
<td>Mean and Variance of $T_{N,n}$</td>
<td>A-21</td>
</tr>
<tr>
<td>A2.3</td>
<td>State Occupancy Probabilities</td>
<td>A-26</td>
</tr>
<tr>
<td>A2.4</td>
<td>System Reliability and Availability</td>
<td>A-27</td>
</tr>
<tr>
<td>A2.4.1</td>
<td>System Reliability</td>
<td>A-27</td>
</tr>
<tr>
<td>A2.4.2</td>
<td>System Availability</td>
<td>A-28</td>
</tr>
<tr>
<td>A2.4.3</td>
<td>Average Availability</td>
<td>A-31</td>
</tr>
<tr>
<td>A2.5</td>
<td>Expected Number of Software, Hardware and Total Failures by Time $t$</td>
<td>A-32</td>
</tr>
<tr>
<td>A2.5.1</td>
<td>Expected Number of Software Failures</td>
<td>A-32</td>
</tr>
<tr>
<td>A2.5.2</td>
<td>Expected Number of Hardware Failures</td>
<td>A-33</td>
</tr>
<tr>
<td>A2.5.3</td>
<td>Expected Number of Total Failures</td>
<td>A-35</td>
</tr>
<tr>
<td>A2.5.4</td>
<td>Illustrative Example</td>
<td>A-37</td>
</tr>
<tr>
<td>A3</td>
<td>Operational Cost Models</td>
<td>A-39</td>
</tr>
<tr>
<td>A3.1</td>
<td>Operational Cost Model: Hardware System</td>
<td>A-40</td>
</tr>
<tr>
<td>A3.2</td>
<td>Operational Cost Model: Software System</td>
<td>A-53</td>
</tr>
<tr>
<td>A3.3</td>
<td>Operational Cost Model: Hardware-Software System</td>
<td>A-67</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------</td>
</tr>
<tr>
<td>1.1</td>
<td>Table of Failure-Rate Based Software Reliability Models</td>
<td>1-27</td>
</tr>
<tr>
<td>1.2</td>
<td>Summary of Failure-Rate Based Models</td>
<td>1-28</td>
</tr>
<tr>
<td>2.1</td>
<td>Software Failure Data from NTDS</td>
<td>2-45</td>
</tr>
<tr>
<td>2.2</td>
<td>Kolmogorov-Smirnov Test for the NTDS Data Set</td>
<td>2-51</td>
</tr>
<tr>
<td>2.3</td>
<td>Software Project Characteristics</td>
<td>2-57</td>
</tr>
<tr>
<td>2.4</td>
<td>Description of the Data Sets</td>
<td>2-60</td>
</tr>
<tr>
<td>2.5</td>
<td>Software Data Sets DS1 to DS4</td>
<td>2-61</td>
</tr>
<tr>
<td>2.6</td>
<td>Data for Kolmogorov-Smirnov Test (Data Set DS1)</td>
<td>2-65</td>
</tr>
<tr>
<td>2.7</td>
<td>A Summary of Data Analyses</td>
<td>2-77</td>
</tr>
<tr>
<td>2.8</td>
<td>Failures in One Hour (Execution Time) Intervals and Cumulative Failures</td>
<td>2-79</td>
</tr>
<tr>
<td>2.9</td>
<td>Error Data Based on Levels of Seriousness</td>
<td>2-93</td>
</tr>
<tr>
<td>2.10</td>
<td>Error Data Based on Types of Errors</td>
<td>2-95</td>
</tr>
<tr>
<td>2.11</td>
<td>Number of Various Types of Errors and Estimates of (a) and (b)</td>
<td>2-104</td>
</tr>
<tr>
<td>2.12</td>
<td>Number of Errors by Severity</td>
<td>2-112</td>
</tr>
<tr>
<td>3.1</td>
<td>Software Failure Data from Thayer et al. [THA76]</td>
<td>3-33</td>
</tr>
<tr>
<td>3.2</td>
<td>A Summary of Data Analysis for DS1 to DS4</td>
<td>3-36</td>
</tr>
<tr>
<td>A2.1</td>
<td>Mean, Variance, and Standard Deviation of First Passage Time Distribution, (N = 10)</td>
<td>A-25</td>
</tr>
<tr>
<td>A2.2</td>
<td>Selected Values of (P_{N,n}(t)) and (A(t)), (N = 10)</td>
<td>A-30</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>A2.3</td>
<td>Expected Number of Failures Detected Per Time</td>
<td>A-36</td>
</tr>
<tr>
<td>A3.1</td>
<td>Average Availability and Expected Number of Failures, Hardware System Failure Rate: 0.010</td>
<td>A-46</td>
</tr>
<tr>
<td>A3.2</td>
<td>Expected Total Cost Per Unit Time, Hardware System Failure Rate: 0.010</td>
<td>A-47</td>
</tr>
<tr>
<td>A3.3</td>
<td>Average Availability and Expected Number of Software Failures, Software System Failure Rate: 0.050</td>
<td>A-60</td>
</tr>
<tr>
<td>A3.4</td>
<td>Expected Total Cost Per Unit Time, Software System Failure Rate: 0.050</td>
<td>A-61</td>
</tr>
<tr>
<td>A3.5</td>
<td>Average Availability and Expected Number of Failures, Hardware-Software System ( t = 100 )</td>
<td>A-71</td>
</tr>
<tr>
<td>A3.6</td>
<td>Average Availability and Expected Number of Failures, Hardware-Software System ( t = 500 )</td>
<td>A-72</td>
</tr>
<tr>
<td>A3.7</td>
<td>Average Availability and Expected Number of Failures, Hardware-Software System ( t = 1000 )</td>
<td>A-73</td>
</tr>
<tr>
<td>A3.8</td>
<td>Average Availability and Expected Number of Failures, Hardware-Software System ( t = 2000 )</td>
<td>A-74</td>
</tr>
<tr>
<td>A3.9</td>
<td>Expected Total Cost Per Unit Time, Hardware-Software System ( C_{S_1} = 10, C_{H_1} = 10, C_{S_2} = 10, C_{H_2} = 10, C_{S_3} = C_{H_3} = 10, t = 100 )</td>
<td>A-75</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1.1</td>
<td>Functional View of Software</td>
<td>1-5</td>
</tr>
<tr>
<td>1.2</td>
<td>Software Error</td>
<td>1-6</td>
</tr>
<tr>
<td>2.1</td>
<td>A Graphical Representation of the Deterministic Model for Software Failure</td>
<td>2-6</td>
</tr>
<tr>
<td>2.2</td>
<td>Log-Likelihood Surface Based on NTDS Data</td>
<td>2-46</td>
</tr>
<tr>
<td>2.3</td>
<td>Plots of Mean Value Function and 90% Confidence Bounds for the N(t) Process (NTDS Data)</td>
<td>2-49</td>
</tr>
<tr>
<td>2.4</td>
<td>95% Confidence Bounds for the Conditional c.d.f. G(x) and the Fitted C.D.F. Curve (NTDS Data)</td>
<td>2-53</td>
</tr>
<tr>
<td>2.5</td>
<td>Joint Confidence Regions for a and b (NTDS Data)</td>
<td>2-55</td>
</tr>
<tr>
<td>2.6</td>
<td>Actual and Expected Cumulative Number of Failures and 90% Confidence Bounds for the N(t) Process for Data Set DS1</td>
<td>2-63</td>
</tr>
<tr>
<td>2.7</td>
<td>95% Confidence Bounds for the Conditional c.d.f. G(t_i) and the Fitted Curve for DS1 Data</td>
<td>2-67</td>
</tr>
<tr>
<td>2.8</td>
<td>Joint Confidence Regions for a and b for Data Set DS1</td>
<td>2-70</td>
</tr>
<tr>
<td>2.9</td>
<td>Expected Number of Remaining Software Errors and Related Quantities for Various t (Data Set DS1)</td>
<td>2-74</td>
</tr>
<tr>
<td>2.10</td>
<td>Reliability and 90% Confidence Bounds After 15 Weeks of Testing</td>
<td>2-76</td>
</tr>
<tr>
<td>2.11</td>
<td>Plot of the Number of Failures Per Hour (SYS1)</td>
<td>2-80</td>
</tr>
<tr>
<td>2.12</td>
<td>Plot of the Number of Failures Per Hour (SYS2)</td>
<td>2-81</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.13</td>
<td>Number of Failures and 90% Confidence Bounds (SYS1).</td>
<td>2-83</td>
</tr>
<tr>
<td>2.14</td>
<td>Number of Failures and 90% Confidence Bounds (SYS2).</td>
<td>2-84</td>
</tr>
<tr>
<td>2.15</td>
<td>Observed and Expected Number of Remaining Errors with 90% Confidence Bounds on E[N(x)] - SYS1</td>
<td>2-85</td>
</tr>
<tr>
<td>2.16</td>
<td>Observed and Expected Number of Remaining Errors with 90% Confidence Bounds on E[N(x)] - SYS2</td>
<td>2-86</td>
</tr>
<tr>
<td>2.17</td>
<td>Reliability and 90% Confidence Bounds - SYS1</td>
<td>2-88</td>
</tr>
<tr>
<td>2.18</td>
<td>Reliability and 90% Confidence Bounds - SYS2</td>
<td>2-89</td>
</tr>
<tr>
<td>2.19</td>
<td>Actual Software Errors with Several Levels of Seriousness During 22-Month Period of Testing.</td>
<td>2-97</td>
</tr>
<tr>
<td>2.20</td>
<td>The Fitted Mean Value Functions for Several Levels of Seriousness.</td>
<td>2-98</td>
</tr>
<tr>
<td>2.21</td>
<td>Actual Software Errors of Several Types During 22-Month Period of Testing.</td>
<td>2-99</td>
</tr>
<tr>
<td>2.22</td>
<td>The Fitted Mean Value Functions for Several Types of Errors.</td>
<td>2-100</td>
</tr>
<tr>
<td>2.23</td>
<td>Likelihood Surface (Total Errors).</td>
<td>2-105</td>
</tr>
<tr>
<td>2.24</td>
<td>Contours of the Likelihood Surface in the (a-b) Plane.</td>
<td>2-106</td>
</tr>
<tr>
<td>2.25</td>
<td>Observed Number of Software Errors (Apollo).</td>
<td>2-107</td>
</tr>
<tr>
<td>2.26</td>
<td>Expected Number of Software Errors (Apollo).</td>
<td>2-108</td>
</tr>
<tr>
<td>2.27</td>
<td>Observed Number of Errors by Severity (Boeing).</td>
<td>2-113</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>------------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>2.28</td>
<td>Expected Number of Errors by Severity.</td>
<td>2-114</td>
</tr>
<tr>
<td>3.1</td>
<td>A Plot of Observed Number of Failures Per Week for Data Set DS1.</td>
<td>3-34</td>
</tr>
<tr>
<td>3.2</td>
<td>Plots of the Cumulative Number of Software Failures for DS1 to DS4.</td>
<td>3-37</td>
</tr>
<tr>
<td>3.3</td>
<td>Plots of the Expected Cumulative Number of Software Failures (m(t)) for DS1 to DS4.</td>
<td>3-38</td>
</tr>
<tr>
<td>3.4</td>
<td>Estimated Mean Value Function and 90% Confidence Bounds for the N(t) Process (DS1).</td>
<td>3-40</td>
</tr>
<tr>
<td>3.5</td>
<td>Estimated 90% Confidence Bounds for E(N(24)) and Values of E(N(t)) (DS1).</td>
<td>3-43</td>
</tr>
<tr>
<td>3.6</td>
<td>Reliability Function and 90% Confidence Bounds (DS1).</td>
<td>3-44</td>
</tr>
<tr>
<td>3.7</td>
<td>Estimated Mean Value Function and 90% Confidence Bounds for the N(t) Process (Data Set NTDS).</td>
<td>3-46</td>
</tr>
<tr>
<td>4.1</td>
<td>Plots of Reliability Versus bs for m(x) = 5(5)50.</td>
<td>4-5</td>
</tr>
<tr>
<td>4.2</td>
<td>Cost Components.</td>
<td>4-9</td>
</tr>
<tr>
<td>4.3</td>
<td>Plots of λ(T) Versus T</td>
<td>4-11</td>
</tr>
<tr>
<td>4.4</td>
<td>Contours of T₀ in the b, c₃/(c₂-c₁) Plane (a = 1348).</td>
<td>4-14</td>
</tr>
<tr>
<td>4.5</td>
<td>Plots of C(T;t) and Λ(T) for γ &gt; 1 and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a(γ⁻¹/y⁺⁻¹/y⁺⁻¹) &lt; C₃/C₂-C₁</td>
<td>4-17</td>
</tr>
<tr>
<td>4.6</td>
<td>Plots of C(T;t) and Λ(T) for γ &gt; 1 and</td>
<td></td>
</tr>
<tr>
<td></td>
<td>a(γ⁻¹/y⁺⁻¹) = C₃/C₂-C₁</td>
<td>4-19</td>
</tr>
</tbody>
</table>
Figure

4.7 Plots of C(T;t) and \( A(T) \) for \( \gamma > 1 \) and
\[
\alpha (\gamma - 1) (\gamma - 1 / \gamma) > \frac{C_3}{C_2 - C_1}
\]
 ........................................ 4-20

4.8 Plots of C(T;t) and \( \mu(T) \) for \( \gamma < 1 \) .................. 4-23

A2.1 Up and Down Behavior of Hardware-Software System ...................... A-5

A2.2 Diagrammatic Representation of Transitions Between States of X(t) ........ A-11

A2.3 A Typical Realization of X(t) Process .................................. A-13

A2.4 Probability Distribution Function of First Passage Time to \( n \) ............ A-23

A2.5 Cumulative Distribution Function of First Passage Time to \( n \) ............ A-24

A2.6 State Occupancy Probabilities and System Availability ......................... A-29

A2.7 Expected Cumulative Number of Failures .................................. A-38

A3.1 Average Availability Vs. Repair Rate for Different Failure Rates (\( t = 500 \)) A-50

A3.2 Expected Total Cost/Unit Time Vs. Repair Rate for Different Cost Factors (\( \beta = .01 \), \( t = 500 \)) .................. A-51

A3.3 Expected Total Cost/Unit Time Vs. Repair Rate for Different Times (\( \beta = .01 \), \( C_{h_1} = 10 \), \( C_{h_2} = 10 \), \( C_{h_3} = 100 \)) .......................... A-52

A3.4 Software System: Diagrammatic Representation of Transitions Between States of X(t) A-54

A3.5 Average Availability Vs. Repair Rate for Different Failure Rates (\( t = 500 \)) A-64

A3.6 Expected Total Cost/Unit Time Vs. Repair Rate for Different Cost Factors (\( \lambda = .05 \), \( t = 500 \)) .................. A-65
Figure

A3.7  Expected Total Cost/Unit Time Vs. Repair Rate for Different Times ($\lambda = .05, C_{s_1} = 10, C_{s_2} = 10, C_{s_3} = 100$) ............................. A-66

A3.8  Contours of Average Availability Vs. Repair Rates: Hardware-Software System ($\beta = .01, \lambda = .05, t = 500$) ............................. A-76

A3.9  Surface of Average Availability Vs. Repair Rates: Hardware-Software System ($\beta = .01, \lambda = .05, t = 500$) ............................. A-77

A3.10 Contours of Expected Total Cost/Unit Time Vs. Repair Rates: Hardware-Software System ($\beta = .01, \lambda = .05, t = 500, \text{Cost Factor} = 10$) ............................. A-78

A3.11 Expected Total Cost/Unit Time Vs. Repair Rates: Hardware-Software System ($\beta = .01, \lambda = .05, t = 500$) ............................. A-79
SECTION 1
INTRODUCTION AND OVERVIEW

An important quality attribute of a computer system is the degree to which it can be depended upon to perform its intended function in the specified environment.

Evaluation and prediction of this attribute has concerned computer designers and users from the early days of their evolution. Until the late sixties, the attention was almost solely on the performance of hardware aspects of the system. In the early seventies, software became the center of attention due to a continuing increase in the ratio of software to hardware costs, in both the production and the operational phases.

The performance of a software system is dependent on the tools and methodologies used during its development, and an important measure of performance is the nature and frequency of software errors.

This report is primarily concerned with the development of stochastic models for describing the software error occurrence phenomenon and determining software reliability. A description of software errors and their sources is given in Section 1.1 and an error classification scheme
is described in Section 1.2. The notion of software reliability is discussed in Section 1.3. Current suggested approaches for enhancing software reliability are given in Section 1.4. A description of software reliability models reported in the literature is given in Section 1.5.

A non-homogeneous Poisson process (NHPP) model based on an exponentially decaying error occurrence rate is developed in Section 2. Many useful software performance measures are developed and several software failure data sets are analyzed to show the applicability and usefulness of this model.

In Section 3, another NHPP model is proposed which can be used to model both the increasing and the decreasing failure rates during the software integration testing phase.

The problem of when to stop testing and start using software is discussed in Section 4. Various useful scenarios are considered and optimum release time policies are developed. The results are illustrated via numerical examples.

A related problem of modelling the total hardware-software system is addressed in Appendix A. This task
was not a part of the research work under the reference contract and the results are reported here as they are considered useful for people interested in software reliability modelling.
1.1 SOFTWARE ERRORS AND THEIR SOURCES

Software (also called program) is essentially an instrument for transforming a discrete set of inputs into a discrete set of outputs (see Figure 1.1). It comprises of a set of coded statements whose function may basically be one of the following:

1. Evaluate an expression and store the result in a temporary or permanent location.
2. Decide which statement to execute next.
3. Perform input/output operations.

Since, to a large extent software is produced by humans, the finished software product is often imperfect. It is imperfect in the sense that a discrepancy exists between what the software can do versus what the user or the computing environment wants it to do. The computing environment refers to the physical machine, operating system, compiler and translators, utilities, etc. These discrepancies are what we call software errors (see Figure 1.2). Basically, the software errors can be attributed to the following:

1. Ignorance of the user requirements;
2. Ignorance of the rules of the computing environment; and
Fig. 1.1. Functional View of Software
Fig. 1.2. Software Error
3. Poor communication of software requirements between the user and the programmer or poor documentation of the software by the programmer. The fact of the matter is even if we know that software contains errors, we may not know with certainty the exact identity of these errors.

Currently, there are two major paths one can follow to expose software errors:

1. Program proving, and
2. Program testing.

Program proving is more formal and mathematical while program testing is more practical and still remains to be heuristic in its approach. The approach in program proving is the construction of a finite sequence of logical statements ending in the statement (usually the output specification statement) to be proved. Each of the logical statements is an axiom or is a statement derived from earlier statements by the application of an inference rule. Program proving making use of inference rules is known as the Inductive Assertion Method. This method was mainly popularized by Floyd, Hoare, Dijkstra and recently Reynolds. Other work on program proving is the work on the Symbolic Execution Method. This method is the basis of some automatic program verifiers. Despite the formalism
and mathematical exactness of program proving, it is still an imperfect tool for verifying program correctness. Gerhart and Yelowitz [GER 76] showed several programs which were proven to be correct but still contained errors. The errors were due to failures in defining what exactly to prove and were not failures of the mechanics of the proof itself.

Program testing is the symbolic or physical execution of a set of test cases with the intent of exposing embedded errors (if any) in the program. Like program proving, program testing remains an imperfect tool for verifying program correctness. A given testing strategy is good for exposing certain kinds of errors but not all possible kinds of errors in a program. An advantage of testing is that it provides accurate information about a program's actual behavior in its actual computing environment; proving is limited to conclusions about the program's behavior in a postulated environment.

Neither proving nor testing can, in practice, guarantee complete confidence on the correctness of programs. Each has its pluses and minuses. They should not be viewed as competing tools. They are, in fact, complementary methods for decreasing the likelihood of program failure [GOO 77].
1.2 SOFTWARE ERROR CLASSIFICATION

A systematic study of software errors in a program requires knowing what specifically these errors are and knowing which tool(s) to use to expose particular types of software errors. Software errors can be grouped as syntax, semantic, runtime, specification and performance errors.

1.2.1 Syntax Errors

These errors are due to discrepancies between the program code and the syntax rules governing the parser or lexical analyzer of a program translator. These are the easiest errors to detect. They can be detected by visual inspection of the code or can be detected mechanically during the program compilation process. Experienced programmers rarely commit syntax errors.

1.2.2 Semantic Errors

These errors are due to discrepancies between the program code and what the semantic analyzer of the computing environment accepts. Among the popular kinds of semantic errors are typechecking errors and implementation restriction errors. Again, they may be detected by the semantic analyzer of a program translator or by visual inspection.
Syntax and semantic errors are detected during the compilation stage of a program. A program having syntax and/or semantic errors cannot be executed. Syntax and semantic errors are mainly due to the ignorance/negligence on the part of the programmer about the restrictions and limitations of the language (s)he is using.

1.2.3 Runtime Errors

As the name implies, runtime errors occur during the actual running of a program. They may be further classified into three categories:

Domain errors

A domain error occurs whenever the value of a program variable exceeds its declared range or exceeds the physical limits of the hardware representing the variable. The declared range of a variable is done implicitly or explicitly. FORTRAN, for example, assigns types to variables based on the variable name or based on a declaration statement. PASCAL requires all variables to be explicitly declared in a declaration statement. PASCAL has facilities to declare ranges by enumeration and/or subsets of numeric domains.

Some program translators produce runtime code for checking certain types of domain errors. Some have built-in recovery features for domain errors (e.g. PL/1, COBOL) and others (e.g. FORTRAN) simply abort execution upon the occurrence of a domain error. Certain compilers, like
PASCAL, automatically check for values outside a declared range.

Domain errors are a serious matter because
a) program execution is aborted, and/or
b) program results are incorrect.

Execution abortion may be fatal especially in real-time systems. Despite their seriousness, domain errors have never been formally and extensively studied in the literature. This is because detection of domain errors can be very difficult. They require exact specification of the ranges of the input variables. Also, the test values required to expose these errors may occur at the input domain's boundary or inside the input domain itself.

**Computational errors**

Computational errors, sometimes known as logic errors, result whenever the program results in an incorrect output. The incorrect output may be due to a wrong formula, an incorrect control flow, assignment to a wrong variable, incorrect parameter passing, etc.

It is not possible to generate runtime code to detect computational errors during program execution. This is because computational errors are really discrepancies between the program's output and the program's specifications.
Computational errors due to incorrect program constructs and statements may be detected by any of the structure dependent or structure independent testing techniques. However, none of these tools can guarantee total absence of these types of computational errors in a program. Computational errors due to missing program constructs and statements may be detected by any of the structure independent testing techniques. Again, none of these tools can guarantee total absence of computational errors due to missing paths.

**Non-Termination errors**

Non-termination error is simply the failure of a program to terminate in finite time without outside intervention. The most common cause of non-termination errors is when the program runs into an infinite loop. Non-termination can also occur if a set of concurrent programs falls into a dead lock.

Infinite loops are detected by simply executing each of the loops in a program. However, this strategy may not guarantee total absence of infinite loops. Some infinite loops may only occur if certain program variables achieve certain values. Program proving may also be used on certain programs to expose infinite loops. The problem of program non-termination in general is still an unsolved problem.
1.2.4 Specification Errors

Specification errors result whenever there exists a discrepancy between the statement of specifications and the statement of user requirements. A requirements error exists whenever there is a discrepancy between the statement of user requirements and the real user requirements.

Presently, detection of specification errors such as:
1. Incomplete specifications,
2. Inconsistent specifications, and
3. Ambiguous specifications,
remains an informal process. This is mainly due to the nonexistence of a specification language powerful enough to translate the user requirements into clear, complete and consistent terms.

A testing tool to detect specification errors is yet to be developed.

1.2.5 Performance Errors

Performance errors exist whenever a discrepancy exists between the actual performance (efficiency) of the programs and its desired or specified performance. Program performance may be measured in a number of ways:

1. Response time
2. Elapsed time
3. Memory space usage
4. Working set requirement, etc.
The actual measurement of the above measures of program performance can be a very difficult process. Program complexity theory tries to estimate bounds on the running time of certain program algorithms. Statistical analysis and simulation can also be employed to estimate the above performance variables. However, use of these tools can be very expensive and time consuming.

A performance testing tool that is economical (time-wise and costwise) to use is yet to be developed.

The most expensive kind of software errors to eliminate are those which are not discovered until late in the software development, such as when the software becomes operational. These are known as persistent software errors. Glass [GLA81] reported that persistent software errors are mostly due to the failure of the problem solution (i.e. the program) to match the complexity of the problem to be solved (i.e. the user requirements). Examples of such errors are computational errors due to missing or insufficient predicates and failure to reset a variable to some baseline value after its use in a functional logic segment. The solution to this software problem is beyond the current state-of-the-art.
1.3 SOFTWARE RELIABILITY

There are a number of conflicting views as to what software reliability really is and how it should be quantified. The conflict arises because of the disagreement in the basic definition of the term "software reliability". Software reliability in the view of some people, especially the computer science purists, should be closely tied to the correctness of software. They argue that an incorrect software (i.e., a software still containing errors) is doomed to fail sooner or later and thus its reliability should be zero (0). Once the software has been freed of all errors, then its reliability becomes one (1). On the other hand, software reliability, as viewed by many engineers, statisticians, and practitioners, should be closely tied to the concept of "probabilistic reliability". These groups of people argue that many programs used in the real world are known to still contain errors and yet they are executed day after day without occurrences of failures. Software reliability, they believe, should be viewed as the probability that a software system will operate without a failure for a specified (mission) time.

One way to resolve this conflict is to look back at the original problem in the real world and ask ourselves the question: "Why do we need to know software reliability?"
The original real world problem, in very simple terms, is as follows:

Develop software that will satisfy the user's requirements in the most efficient (in both time and money sense) way possible.

The solution to this problem turns out to be very difficult basically because of the following facts:

1. Real world software is large and complex.
2. Users are not always 100 percent certain about their requirements,
3. Resources (time and money) allocated for software development are always limited.

Even if we know that we only need, say, 2000 test cases to run for exposure all possible embedded errors in a software, chances are that, in the real world, we may not have enough time and money to perform this exhaustive test. As more and more errors are uncovered by our testing or correctness verification process, the additional cost of exposing the other remaining errors rises very fast. Thus, beyond a point it is almost practically useless to continue testing to achieve 100 percent correctness. This explains the reason why most all software systems in public and private use still have embedded errors.
If we adopt the point of view of a computer science purist, then almost all software systems in use (including those that are accepted as very reliable and useful by their users) have zero reliability. Since everything now has zero reliability, the value or usefulness of the software reliability concept is lost.

The reason why people introduced the concept of software reliability (or hardware reliability for that matter) is to have a useful measure that may help us in dealing with the original real-world software (hardware) problem. This measure is useful in planning and controlling additional resources (time and money) for enhancing the reliability of a software. It is also a useful measure for giving the user confidence about the software quality.

Should we, then, adopt the hardware-based concept of software reliability? One answer to this question at this point in time is yes, but with extreme care. We should be careful because there are inherent differences between software and hardware. Hardware exhibits mixtures of decreasing and increasing failure rates. The decreasing failure rate is due to the fact that as use time on the hardware system accumulates, more and more errors (most probably design errors) are encountered and fixed. The increasing failure rate is due primarily to hardware component
wearout. There is no such thing as wearout in software. It is true that software may become obsolete because of changes in the user and computing environment but once we modify software to reflect these changes, then we are no longer talking of the same software but an enhanced or modified version. Like hardware, software exhibits a decreasing failure rate as the usage time on the system accumulates and errors (due to design and coding) are fixed. Thus, a hardware-type approach to software reliability should be done only in appropriate environments.

Suppose we declare that the reliability of a given software is 0.95. What does this number exactly mean? Following the probabilistic point of view, this may mean any one of the following:

1. If we execute the software several times, 95 percent of the time it will give correct results.
2. We are 95 percent confident that the software will give correct results when executed.

The first interpretation is the so-called frequency interpretation and the second is the so-called subjective interpretation. Littlewood's contention [LIT80] is that in the absence of a "scientific" verifiable meaning for the number 0.95, the only reasonable interpretation is the subjective interpretation.
The only problem we see with this number is its possible inconsistency. A software may have been declared 95 percent reliable by the developer but may have a different perceived reliability by the user and probably a different perceived reliability by another user. A very simple example will illustrate this point. Suppose a software is composed of 100 modules. Because of practical considerations, the software developer stops testing after 90 modules. He then declares the reliability of the system as 90 percent. A user buys the system and happens to use in his particular application some modules (or maybe program paths) which have not been tested. As a result, 50 percent of the time, the user gets incorrect results. His perceived reliability of the system is therefore 50 percent. Another user might use a different mixture of untested modules (program paths) and might get a different number for the reliability measure. The basic question is: What is the true reliability of the software?

The only way to resolve this question, we feel, is to further qualify or condition our software reliability measure. Ultimately, what is more important is that the user gets his correct results from the software. Thus, that user should be more concerned about a reliability measure conditioned with respect to his requirements. The software developer should be concerned with a reliability measure conditioned with respect to the intended specifica-
tions of the system. We should remember that the purpose of the reliability measure is to help in planning and controlling the production of software and nothing more. We may pool all the users into one big user (for example, user of an operating system software) and come up with an average reliability measure. Still, this number may not match the developer's measured reliability. If we let $R[S|r]$ mean the reliability of the software system $S$ with respect to requirements $r$, then in general, we have:

$$R[S|\text{user requirements}] \neq R[S|\text{developer requirements}]$$
1.4 APPROACHES FOR ENHANCING SOFTWARE RELIABILITY

Consider the concept of software reliability based on the following definition [MYE76]:

Software reliability is the probability that the software will execute for a particular period of time without a failure, weighted by the cost to the user of each failure encountered.

This definition is not necessarily based on the actual number of errors residing within the software system but is based on the impact that the errors have on the users. For example, a single error in a space shuttle control software is much more important than several errors in a matrix inversion software system which cause only "trivial" failures. Certainly, a software system which does not contain a serious error but has many trivial errors would generally be considered much more reliable than a system which does not contain the trivial errors but contains the single serious error.

The reliability of a software system is generally expected to grow as it evolves from the design stage to the coding stage and testing stages and down to the operational and maintenance stage. Modern software engineering practice advocates that testing should be performed as early as the design stage. Software errors detected in the design stage are easier and less expensive to remove than those detected during the testing or operational
stage. We also know that modern software design methodologies help in the likelihood of not committing errors in the design stage. Redundant programming, that is, implementing a software in different ways, is sometimes used to enhance software reliability. Fault tolerance programming is another popular technique. However, testing still remains the most commonly used approach to enhancing software reliability.

Testing for the presence of errors is usually done in stages [CHA78]:

1. First stage is the testing done at the module level by the implementing programmer.
2. Modules are then integrated forming a subsystem or the whole system is tested. The system is then tested. This is also known as alpha testing.
3. The software is then given to several "friendly users" who are willing to use the software in an operational environment and the problems encountered with the software are reported. This is known as beta testing.
4. Finally, software is released to all users and corrections are issued against it as problems are reported by the users.

This overall testing process coupled with the design testing process would, hopefully, result in an enhanced
reliability of the software system. Can the reliability of the software decrease as a result of the software correction (debugging) process? The answer is yes. This occurs when additional errors are accidentally injected into the system while removing some other errors.

Hopefully, with the use of better design methodologies, better documentation techniques, better programming languages, better testing strategies and better software management techniques, the likelihood of software reliability decreasing as the system evolves from the design to the operational stage will become less.
1.5 SOFTWARE RELIABILITY MODELS

Many studies have been undertaken during the last decade to analyze and study software failure data with the objective of finding ways that will lead to improved software performance. Such studies can be classified into one (or both) of two categories. In the first category, the emphasis is on the analysis of software failure data collected from small or large projects during development and/or operational phases. Studies in the second category are primarily aimed at the development of analytical models which are then used to obtain the reliability and other quantitative measures of software performance.

The analytical modelling work can then be classified into the following three major categories. The first one emphasizes the stochastic nature of software failures, while the second and the third use combinatorial analysis to provide measures of software reliability,

1. Failure Rate Based Models.
2. Combinatorial or Error-Seeding Models.
3. Input Domain Based Models.
1. **Failure (Hazard) Rate Based Models**: The times between indigenous errors or the number of indigenous errors observed during testing are used to estimate the shape of the hypothesized hazard function. From the estimated hazard function, one can estimate the number of errors remaining in the software, the mean-time-to-failure (MTTF) or the reliability of the software.

2. **Combinatorial or Error Seeding Models**: A known number of errors are seeded (planted) in the program. After testing, the number of exposed seeded and indigenous errors are counted. Using combinatorics and maximum likelihood estimation, estimates of the number of indigenous errors in the program or the reliability of the software can be estimated.

3. **Input Domain Based Models**: The basic approach here is to generate a set of test cases from an input (operational) distribution. Because of the difficulty in estimating the input distribution, the various models in this group partition the input domain into a set of equivalence classes. An equivalence class is usually associated with a program path. The reliability measure is calculated from the observed failures after execution (symbolic or physical) of the sampled test cases.
1.5.1 **Failure Rate Based Models**

Failure rate based models can be further classified as shown in Table 1.1.

The failure-rate (also known as hazard rate) function \( z(t) \) is defined as the conditional probability that an error is exposed in the interval \( t \) to \( t+\Delta t \), given that the error did not occur prior to time \( t \) [MYE76]. The reliability function \( R(t) \) is the probability that no errors will occur from time zero to time \( t \). Reliability theory tells us that \( z(t) \) and \( R(t) \) are related in the following form:

\[
\frac{dz(t)}{dt} = \frac{-dR(t)/dt}{R(t)}
\]

or

\[
R(t) = \exp\left(- \int_0^t z(x)dx\right)
\]

Also, mean-time-to-failure (MTTF) = \( 1/z(t) \).

Estimation of reliability, once the failure rate function \( z(t) \) is known is thus straightforward. The failure rate based models given in Table 1.1 basically differ in their assumption on the failure rate function \( z(t) \). Table 1.2 below displays these differences;
### TABLE 1.1  TABLE OF FAILURE-RATED BASED SOFTWARE RELIABILITY MODELS

<table>
<thead>
<tr>
<th>Error-Count Based Failure Rate Models</th>
<th>Classical</th>
<th>Bayesian</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>De-Eutrophication Process Model of Jelinski-Moranda [JEL72]</td>
<td>Littlewood Model [LIT80]</td>
</tr>
<tr>
<td></td>
<td>Linear Function Testing Time Model of Schick and Wolverton [SCH78]</td>
<td>Goel-Okumoto Imperfect Debugging Model [GOE79]</td>
</tr>
<tr>
<td></td>
<td>Parabolic Function Testing Time Model of Schick and Wolverton [SCH78]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shooman Model [SHO72]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shooman-Natarajan Model [DUN82]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Execution Time Model of Musa [MUS75]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Non-Homogeneous Poisson Process Model of Goel &amp; Okumoto [GOE79]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Geometric De-Eutrophication Process Model of Moranda [MOR75]</td>
<td>Littlewood and Verrall Model [LIT73]</td>
</tr>
<tr>
<td></td>
<td>Geometric Poisson Process Model of Moranda [MOR75]</td>
<td>Thompson &amp; Chelson Model [DUN82]</td>
</tr>
<tr>
<td></td>
<td>Wagoner Model [DUN82]</td>
<td></td>
</tr>
</tbody>
</table>

1-27
<table>
<thead>
<tr>
<th>Model</th>
<th>Assumption on ( z(t) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>De-Eutrophication Process Model</td>
<td>The software failure occurrence rate at any time ( t ) is assumed proportional to the number of errors remaining in the software, i.e., for the time interval between ( (i-1) )st and ( i )th failure, we have ( z(X_i) = \phi[N - (i-1)] ). ( N ) is the initial error content.</td>
</tr>
<tr>
<td>Schick-Wolverton Linear Failure Rate Model</td>
<td>Failure rate is assumed proportional to number of remaining errors in software and test time. For ( i )th interval, ( z(X_i) = \phi[N - (i-1)]X_i ).</td>
</tr>
<tr>
<td>Schick-Wolverton Parabolic Failure-Rate Model</td>
<td>Failure rate is assumed proportional to residual errors and a parabolic function of test time. For ( i )th interval, ( z(X_i) = \phi[N - (i-1)](-ax_i^2 + bx_i + c) ), ( a, b, c &gt; 0 ).</td>
</tr>
<tr>
<td>Shooman Model</td>
<td>( z(t) = K[E_T/I_T - \int_0^T \rho(x)dx] ) where:</td>
</tr>
<tr>
<td></td>
<td>( K ): proportionality constant</td>
</tr>
<tr>
<td></td>
<td>( E_T ): total # errors</td>
</tr>
<tr>
<td></td>
<td>( I_T ): total # instructions (object code)</td>
</tr>
<tr>
<td></td>
<td>( T ): debugging time</td>
</tr>
<tr>
<td></td>
<td>( \rho(x) ): number of errors/total # instruction ( s/x ) debugging time.</td>
</tr>
<tr>
<td></td>
<td>( \int_0^T \rho(x)dx ): total # of errors per ( I_T ) removed during ( T ) time units of debugging time.</td>
</tr>
<tr>
<td>Shooman-Natarajan Model</td>
<td>( z(t) = K\epsilon_r(t) ) where:</td>
</tr>
<tr>
<td></td>
<td>( \epsilon_r ): number of remaining software faults.</td>
</tr>
<tr>
<td></td>
<td>( K ): constant of proportionality</td>
</tr>
</tbody>
</table>
Execution Time
Model of Musa

I. \( z(\tau) = KfN_0 - Kfn(\tau) \) where:

- \( K \) : error exposure ratio
- \( f \) : linear execution frequency of program
- \( N_0 \) : initial error content
- \( \tau \) : CPU time utilized in operating the program
- \( n(\tau) \) : net number of errors corrected during \( \tau \),

II. If \( dn(\tau)/dt = \) error exposure rate, then \( Z(\tau) = KfN_0 \exp[-Kf\tau] \)

Non-Homogeneous Poisson Process (NHPP) Model of Goel & Okumoto

 Assumes that the error detection rate \( \lambda(t) \) is time dependent and is given by:

\[ \lambda(t) = a \exp[-bt] \] where:

- \( a \) : expected number of errors to be eventually detected
- \( b \) : error detection rate per error

Geometric De-Eutrophication Process Model of Moranda

Assume that the steps representing the decrease in failure rate between adjacent failure time are geometrically varying.

\[ Z(X_i) = Dk^{i-1} \] where:

- \( D \) : initial error detection rate
- \( Dk \) : error detection rate after the occurrence of the 1st error
- \( Dk^{i-1} \) : error detection rate after the occurrence of the \( i \)th error.

Geometric Poisson Process Model of Moranda

A superposition of a geometric De-Eutrophication process and a Poisson process with parameter \( \theta \),

\[ Z(X_i) = Dk^{i-1} + \theta \]

Wagoner (Weibull) Model

\[ z(t) = \frac{\lambda}{\sigma} \left( \frac{t}{\sigma} \right)^{\lambda-1} \] where:

- \( \sigma \) : scale parameter
- \( \lambda \) : parameter to squeeze or stretch the shape of the distribution
- \( t \) : CPU time units
Goel-Okumoto
Imperfect Debugging
Model

The form of $Z(t)$ is not obvious: however, the reliability function between the $(k-1)^{st}$ and $K^{th}$ failure is:

$$R_K(x) = \sum_{j=0}^{K-1} (K-1)P^{K-j-1}Q^j \exp[-(N-K+j+1)x]$$

where:

- $P$: Prob (of successful correction of a defect)
- $Q$: Prob (of imperfect debugging) = 1 - $P$
- $\lambda$: Parameter of the exponential distributions governing the time between failures
- $N$: Estimated initial number of defects.

Thompson & Chelson
Model

$z(t) = \lambda$ but $\lambda$ is treated as a random variable with a gamma density function

$$T_0(\lambda T_0)^{K_0} \frac{T_0}{T(K_0 + 1)} \exp[-T_0]$$

where:

- $K_0$: observed failures
- $T_0$: testing time for $K_0$
- $K_0$ and $T_0$ essentially represent previous testing experience.

Littlewood and
Verrall Model

$z(t) = \lambda$ but $\lambda$ is treated as a random variable distributed as Gamma with shape parameter $\alpha$ and scale parameter $\psi(i)$, an increasing function of $i$.

Littlewood Model

$Z(X_i) = \lambda_i$ and $\lambda_i$ is distributed as $gamma((N-i+1)\alpha, \beta + \sum t_j)$, where:

- $N-i+1$: number of errors remaining when $(i-1)$ failures have occurred.
- $t_j$: execution time from $(j-1)^{st}$ failure to the $j^{th}$ failure.
- $\alpha, \beta$: parameters of gamma distribution.
As noted above a number of assumptions are made in the development of failure-rate models. A discussion of these assumptions is provided in the following paragraphs to point out the dangers in the use of these models when the assumptions are not satisfied. It should, however, be noted that some models could be robust to departures from many of these assumptions and can be used for reliability assessment purposes.

1. All the models described above assume that any error detected is immediately corrected. The correction process does not alter the program. All corrections remove the detected error (except the IDM model) and do not result in the introduction of new errors. It is not hard to accept that correction of a detected error in a program may result in new errors in the program. Goel and Okumoto [GOE79] tried to address the second limitation above by formulating the Imperfect Debugging Method (IDM). IDM assumes that the number of errors in the system at time t is governed by a Markov process. Time between transitions is exponentially distributed with rates dependent on the current error content of the program. The state transitions are governed by the probability of imperfect debugging. No one has yet addressed the problem in which the debugging process introduces new errors into the software.
2. Models such as those by Jelinski and Moranda, Musa, and Shooman assume that the software failure rate is a constant multiple of the number of remaining errors. This is the same as saying that each error in a given time interval (between failures) has the same chance of being detected. This, obviously, is not always true since errors that happen to reside in a portion of the code that is frequently executed by the user (or tested by the user) have a higher probability of being detected. Errors which reside in the unreachable (or never used) portion of the code will obviously have a lower (or zero) probability of being detected. Moranda tried to address this problem by reformulating the De-Eutrophication model into the Geometric De-Eutrophication Model and later to the Geometric Poisson Model. In these variations, the failure rate between adjacent failure intervals is geometrically varying.

The NHPP model by Goel and Okumoto also tried to alleviate these problems (i.e. problems with the constant failure rate models) by postulating a time dependent error detection rate model. Littlewood is more ambitious in trying to address this problem. He postulated a model which assumes that each remaining error in the program has a different rate of occurrence. The failure rate of the overall program is then just the sum of the individual error's rate of occurrence.
3. The Schick-Wolverton models happen to model a process where there is an increasing failure rate between failures. This may be a ridiculous assumption if we argue that software does not wear out. But there can be cases where the software failure rate might in fact increase and this may be attributed to the increased intensity of testing. This phenomenon is usually observed during the early stages of the software development cycle.

4. Basing the time between failures in terms of execution (CPU) time, as was assumed by Musa, Littlewood and Wagoner, may sometimes be unrealistic. An increase in accumulated time between two adjacent failures may not necessarily mean that the software has less and less number of errors or, putting it equivalently, that the software's reliability is improving. A very simple example will illustrate this point. Consider a program containing only a single error. The same copy of the program is given to two debuggers. One debugger spends a lot of time running and re-running the program (which can be very tempting to do on on-line and timesharing systems) trying to uncover the error. The second debugger, on the other hand, spends a lot of time analyzing the program before even attempting to make a test run. Suppose both debuggers are successful in finding the error. What is the resulting reliability of the software?
Execution time theory says that since the CPU time between failures of the first software is larger than that of the second software, then the first software is more reliable. Of course we know that this is not true since both software have the same reliability. Another example where execution time may be misleading is when a selected subset of the program is executed repeatedly. While the execution time is accumulating the test coverage is not and this will lead to an incorrect assessment of reliability. What is the most appropriate time unit to use for interfailure times is still a controversial topic.

5. What about the assumption of independence of interfailure times? Is this a realistic assumption? Chances are it is not. The testing process that is used to uncover errors is usually not a random process. The time to the next failure may very well depend on the nature and time to failure of the previous error. If the previous error was a very critical one, then we might decide to intensify the testing process and look for more critical errors. This intensification in the testing process may mean a shorter time to the next failure than what might have happened if the testing intensity were maintained at normal levels.

6. Most of the models require time between failure data to estimate reliability. There can be cases when the mean
time between failure is infinite; as such, these models become useless. The mean time between failures can be infinite if the user of the software has requirements that would only traverse the error free paths of the program.

7. Basing the reliability of the software on the remaining number of errors can also sometimes be ridiculous. A user does not really care whether a software has a certain number of remaining errors. As long as all his requirements are met correctly by the software, then as far as the user is concerned, the software is 100 percent reliable. Littlewood [LIT80] argued that a program with two bugs in little exercised portions of the code can be more reliable than a program with only one but frequently encountered bug.

8. All the models implicitly assume that the testing process, which generated the estimate for the failure rate, will be the same as the operating environment. This again is not true. A reliability measure conditioned on the user requirements rather than a simple unconditioned software reliability measure would be more realistic.

9. Some models assume that software reliability is time-dependent. Most software fail not because of the length of time it has been in use but fail because of the nature of the input to which it is subjected. Some software like real time control software or
operating systems show an illusion of failing with time-of-use because they are used almost continuously. In those environments, the time-dependence assumption may be valid.

10. Perhaps the most fundamental assumption is the treatment of the software as a black box. At least some software reliability models should take into consideration software characteristics and the characteristics of the software development process in addition to the failure times and the number of remaining errors.
1.5.2 Combinatorial or Error-Seeding Models

A number of combinatorial models have been proposed but the most popular (and most basic) is Mill's Hypergeometric Model. This model requires that a number of known errors be randomly inserted (seeded) in the program to be tested. The program is then tested for some amount of time. The number of original indigenous errors can be estimated from the numbers of indigenous and seeded errors uncovered during the test by using the hypergeometric distribution.

Let

- \( n_0 \) = number of seeded errors
- \( K \) = number of seeded errors detected during testing
- \( N \) = total number of indigenous errors
- \( r \) = number of indigenous errors detected during testing

We then have

\[
P[K \text{ seeded errors in } r \text{ detected}] = \frac{n_0 (N-n_0)}{N r (r-K)}
\]

MLE for \( N = \frac{nr}{k} \)

A variant of the above model is the so-called Binomial model. Let \( q_i \) = Prob [errors] on each run i, then we have

\[
\text{Prob}[x \text{ errors in } y \text{ trials}] = \binom{y}{x} q_i^x (1-q_i)^{y-x}
\]

The serious assumption of the above models is that the indigenous and seeded errors are assumed to have the same probability of being detected. In other words, the seeded errors must be of the same type and should have
the same distribution as the indigenous errors. This, of course, is difficult to meet in real-world conditions.

A suggestion is given in [H078] to overcome this problem. In this improved approach, two teams are going to test the program independently. Suppose team 1 detects \( n \) errors and team 2 detects \( r \) errors, and the number of errors common to both teams is \( K \). We can then view the errors detected by one team, say team 1, as seeded errors, and estimate the total number of indigenous errors \( N \) to be \( nr/K \). Note, however, that simple errors may be discovered first and the distribution of errors detected may not resemble the actual distribution of errors; so that the estimates may be biased.

The advantage that is obvious with these combinatorial models over the failure-rate based models is that they are based on less and much simpler assumptions.

1.5.3 Input Domain Based Models

A good representative set of models in this group includes the Nelson (TRW) model [BRO75], Ho Model [HO78], and the Bastani model [BAS80].

**Nelson (TRW) Model**

The reliability of the software is measured by exposing (running) the software with a sample of \( n \) inputs. The \( n \) inputs are randomly chosen from the input domain set 

\[ E = \{E_i : i = 1, N\} \]

where each \( E_i \) is the set of data values needed to make a run. The random sampling of \( n \) inputs is done according to a probability distribution \( P_i \); the set 

\[ (P_i : i = 1, N) \]

is the "operational profile" or simply user...
input distribution. If $n_e$ is the number of inputs that resulted in execution failure, then an unbiased estimate for the software reliability $\hat{R}$ is $1 - (n_e/n)$. However, it may be the case that the test set used during the verification phase may not be representative of the expected operational usage. Brown and Lipow [BR075] suggested an alternative formula for $\hat{R}$ which is

$$\hat{R} = 1 - \sum_{i=1}^{N} \frac{f_j}{n_j} P(E_j)$$

where

- $n_j = \text{number of runs sampled from input subdomain } E_j$
- $f_j = \text{number of failures observed out of } n_j \text{ runs.}$

The main difference between Nelson's $\hat{R}$ and Brown and Lipow's $\hat{R}$ is that the former explicitly incorporates the usage distribution or the test case distribution while the latter implicitly assumes that the accomplished testing is representative of the expected usage distribution. Both models assume prior knowledge of the operational usage distribution. This may not be easy to do for some real-world software. Another criticism of this approach is the use of random testing.

**Ho Model**

Reliability estimation in this model proceeds by first generating the symbolic execution tree of the program. This tree characterizes all the execution paths and their associated outputs in the program. The nodes represent statements while the edges represent the state vector resulting from symbolic execution along the path.
from the root statement to the current statement. A procedure for generating the symbolic execution tree is given in [H078]:

I. The first statement is the root of the tree.

II. If a leaf is not a STOP or RETURN statement, symbolically execute the statement corresponding to the node. If the current statement is a conditional statement, the feasibility of the branches is examined. New nodes are created for statements which are successors of the current statement. Edges, labelled with state vectors are joined between the current node and the new node(s).

III. Go to II.

The generated execution paths from the symbolic execution tree are proven correct or are sample tested. For a given path, say path i, if it is proven correct, then the path reliability $R_i = 1$. If path i cannot be proven correct, a random sample of N test cases is generated that will execute path i. If no failures result from the execution of the N test cases, then $R_i$ is bounded below by $1 - C_i$ where $C_i$ is the confidence interval of path i. The length of $C_i$ is a function of our given confidence coefficient $\alpha$. On the other hand, if $n$
failures are observed and the errors not corrected, then $R_i$ is bounded below by $\frac{N - n}{N} - C_i$. If the observed $n$ failures are corrected, then the sample testing is repeated for path $i$.

Finally, the software reliability estimate $R$ is calculated from

$$R = \sum_{i=1}^{m} f_i R_i$$

where:

$f_i$ = weighting factor or path $i$ which corresponds to the execution frequency of path $i$.

$m$ = total number of execution paths.

One difficulty with applying this approach is the large number of paths that may exist for real world software.

**Bastani Model**

This input domain based model estimates the reliability $R$ from the relation

$$\hat{R} = 1 - \hat{V}_{e_R}$$

where:

$\hat{V}_{e_R}$ = the total error size remaining in the program.

$\hat{V}_{e_R}$ can be determined by testing the program and locating and estimating the size of errors found [BAS80]. An error
has a large size if it is easily detected (i.e., if it affects many input elements). An error has a small size if it is relatively difficult to detect. The size of an error depends on the way test inputs are selected. Good test case selection strategies like path testing, boundary value analysis, magnify the size of an error since they exercise error-prone constructs. The observed error size is lower if random testing is employed. Although the model does not assume random testing (in fact, any test strategy can be employed), it offers no easy or systematic way to estimate $V_{er}$. 
SECTION 2

A TIME DEPENDENT FAULT DETECTION RATE MODEL

2.1 INTRODUCTION

In this section, our objective is to develop a parsimonious model whose parameters have a physical interpretation, and which can be used to predict various quantitative measures for software performance assessment. Also of interest is the applicability of the model over a broad class of projects. Further, it should be possible to estimate the parameters of the model from available failure data which could be given as either the number of failures in specified time intervals, or as times between software failures.

With this objective, we develop and investigate a nonhomogeneous Poisson process (NHPP) [BR072] model with a time dependent fault detection rate for the software failure phenomenon. By studying the behavior of the cumulative number of failures by time \( t \) process, \( N(t) \) it is shown in section 2.2 that this process can be well described by a non-homogeneous Poisson process (NHPP) with a two parameter exponentially decaying fault detection rate.
NHPP has been used by many researchers to describe random phenomena in various applications [CRO74, DUN75, DVA64]. Some such applications are the occurrences of coal mining disasters [MAG52]; equipment failures [DUA64, LEW64, PRO63]; transactions in a data-base system [LEW76]; and software error counts over a series of time intervals [SCH75]. Various forms of the intensity function for the NHPP used in actual applications are the exponential polynomial rate function [LEW76], a log-linear rate function [COX66], and a Weibull rate function [CRO74, DON75, MOE76].

Several measures for software performance assessment, such as the number of faults remaining in the system, distribution of time to next failure, and software reliability, are proposed in section 2.3. Based on the NHPP model, expressions are then derived for obtaining the estimates and confidence limits for these performance measures.

Two methods are described in section 2.4 for estimating the parameters of the model from available failure data. The first one is for the case when data is given in the form of number of failures in given time intervals. The time intervals can be of equal or un-
equal lengths, but the data must be converted to an interval of the same length. The second method is used when times between software failures are given. Analyses of actual failure data are presented in section 2.5.
2.2 MODEL DEVELOPMENT

A software system in use is subject to failures caused by faults present in the system. The faults are encountered when a sequence of instructions is executed which, in turn, depends on the input data set. In this section, we develop a model to describe this failure occurrence phenomenon.

2.2.1 Deterministic Analysis of Software Failure Process

It is useful to first make a simpler analysis by ignoring the statistical fluctuations in the number of software failures before analyzing the failure phenomenon as a stochastic process [COX65]. Let \( n(t) \) denote the cumulative number of software failures detected by time \( t \). Assume that \( n(t) \) is large enough so that it can be expressed as a continuous function of \( t \). Since the number of errors in a system is a finite value, \( n(t) \) is a bounded non-decreasing function of \( t \) with

\[
    n(0) = 0
\]

and

\[
    n(\infty) = a
\]

(2.1)
For purposes of modeling, we assume that the usage of the system is basically similar over time. Then the number of failures in \((t, t+\Delta t)\) is proportional to the number of undetected faults at \(t\), i.e.,

\[
n(t+\Delta t) - n(t) = b(a-n(t))\Delta t,
\]

where \(b\) is a proportionality constant.

A graphical representation of the above description is provided in Figure 2.1.

Now, from Equation (2.2), we get the differential equation

\[
n'(t) = ab - bn(t).
\]

Taking the Laplace transform [ABR65, BUC56] of Equation (2.3) under the conditions of Equation (2.1), we have

\[
s\hat{n}(s) = \frac{ab}{s} - b\hat{n}(s),
\]

or

\[
\hat{n}(s) = \frac{ab}{s(s+b)},
\]

(2.4)
where

\[ \hat{n}(s) = \int_0^\infty e^{-st} \, dn(t). \] (2.5)

The solution of Equation (2.3) is thus obtained by inverting Equation (2.4) and is given by

\[ n(t) = a(1 - e^{-bt}). \] (2.6)

Under the assumptions discussed above, Equation (2.6) is the deterministic model of the software failure process. For given \( a \) and \( b \), we can easily compute the number of failures to be encountered by some time \( t \) so that the failure phenomenon can be described with certainty. It should be noted, however, that the actual failure phenomenon is not deterministic.

2.2.2 Stochastic Analysis of Software Failure Process

In an actual usage, the software system is subjected to random inputs causing the failures to occur at random times, i.e., the failure phenomenon is stochastic (non-deterministic). Therefore, a realistic description of the failure process must incorporate this randomness.
Let \((N(t), t \geq 0)\) be a counting process \([\text{PYK61, ROS76, SNY75}]\) representing the cumulative number of failures by time \(t\). (Note that \(N(t)\) is a random variable while \(n(t)\) above was taken to be deterministic.) Assuming that each failure is caused by one fault, \(N(t)\) also represents the cumulative number of faults detected by time \(t\). It should be pointed out that a detected fault may not be removed and, as a result, may cause additional failure(s) at a later stage. For the \(N(t)\) process, such recurrences are counted as new events.

Let \(m(t)\) be the mean value function of the \(N(t)\) process, i.e.,

\[
m(t) \equiv E[N(t)] .
\]  

(2.7)

Since \(m(t)\) represents the expected number of software failures or detected faults by time \(t\), it is a non-decreasing function of \(t\). If we assume that there will be a finite number of faults to be detected over a long period of time, \(m(t)\) has the following boundary conditions:

\[
m(t) = \begin{cases} 
0, & t = 0 \\ 
a, & t = \infty
\end{cases}
\]  

(2.8)
where \( a < \infty \) and represents the expected number of software faults to be eventually detected. Furthermore, it is assumed that, for small \( \Delta t \), the expected number of software failures during \((t, t+\Delta t)\) is proportional to the expected number of software faults undetected by time \( t \), i.e.,

\[
m(t+\Delta t) - m(t) = b(a-m(t))\Delta t ,
\]

(2.9)

where \( b \) is a constant of proportionality. Solving the differential equation obtained from Equation (2.9) under the boundary conditions of Equation (2.8), we get

\[
m(t) = a(1 - e^{-bt}) .
\]

(2.10)

This equation specifies the mean value function for the underlying software failure counting process \( N(t) \). The intensity function, obtained by taking the derivative of \( m(t) \), represents the fault detection rate at time \( t \) and is given by

\[
\lambda(t) \equiv m'(t) = abe^{-bt} .
\]

(2.11)
We now study the probabilistic behavior of the \( N(t) \) process by using \( m(t) \) and \( \lambda(t) \). Since there are no failures at \( t = 0 \), we have \( N(0) = 0 \). It is also reasonable to assume that the number of software failures during non-overlapping time intervals are independent. In other words, for any finite collection of times \( t_1 < t_2 < \ldots < t_n \), the \( n \) random variables \( N(t_1), (N(t_2)-N(t_1)), \ldots, (N(t_n)-N(t_{n-1})) \) are statistically independent. This implies that the counting process \( \{N(t), t \geq 0\} \) has independent increments.

We assign the probabilities on the increments of the \( N(t) \) process as follows.

\[
N(t+\Delta t)-N(t) = \begin{cases} 
0 \text{ with probability } 1-\lambda(t)\Delta t+O(\Delta t) \\
1 \text{ with probability } \lambda(t)\Delta t+O(\Delta t) \\
2 \text{ with probability } O(\Delta t) \\
\vdots \\
\vdots 
\end{cases} 
\tag{2.12}
\]

where

\[
\frac{O(\Delta t)}{\Delta t} \to 0 \text{ as } \Delta t \to 0.
\]
The underlying $N(t)$ process satisfying conditions of Equation (2.12) is now a NHPP with mean value function $m(t)$ and intensity function $\lambda(t)$ as given in Equations (2.10) and (2.11), respectively [FELW57, FELW60]. Hence, the distribution of $N(t)$ is given by

$$P(N(t) = y) = \frac{(m(t))^y}{y!} e^{-m(t)}, \quad y = 0, 1, 2, \ldots$$

(2.13)

Under the assumptions discussed above, the stochastic behavior of the software failure phenomenon can be completely described by Equation (2.13). It should be pointed out that Equation (2.9) implies that the ratio

$$\frac{\text{Number of faults detected during } (t, t+\Delta t)}{\text{Number of faults undetected by } t)\Delta t} = b$$

(2.14)

is constant at an; time $t$. Therefore, $b$ can be interpreted as the error detection rate per error.

Equations (2.10) and (2.13) constitute the basic software failure model under study in this report.
2.3 SOFTWARE PERFORMANCE MEASURES

The model developed in section 2.2 is a description of the failure phenomenon. In order to use this model to predict software performance, we generally need expressions for quantitative measures, such as the number of failures by some prespecified time, the number of faults remaining in the software at a future time, and software reliability during a mission. In this section, we develop models that can be employed to estimate such quantities.

2.3.1 Number of Software Faults Detected by t

For given a and b, the distribution of N(t), the cumulative number of software failures detected by time t, is obtained from Equations (2.10) and (2.13) as

\[
P(N(t)=y) = \frac{(a(1-e^{-bt}))^y}{y!} \cdot e^{-a(1-e^{-bt})},
\]

\[y = 0,1,2,\ldots.
\]

In other words, N(t) has a Poisson distribution with mean

\[
m(t) = E[N(t)] = a(1 - e^{-bt}).
\]

(2.15)
Note that

\[ P\{N(\infty) = y\} = \frac{a^y}{y!} e^{-a}, \quad y = 0, 1, 2, \ldots \]  

(2.17)

i.e., the distribution of \( N(\infty) \), the total number of failures encountered or faults detected if the system is used indefinitely, is also a Poisson distribution with mean 'a'. This result is consistent with theoretical studies which indicate that the Poisson process is the limiting distribution of many phenomena similar to the software error occurrence phenomenon [MIL76, SNY75].

2.3.2 **Number of Remaining Faults**

We have been considering the number of failures encountered by time \( t \), \( N(t) \). Since many of the performance measures depend on the number of faults remaining in the system, we now consider this phenomenon.

Let \( \bar{N}(t) \) be the number of faults remaining in the system at time \( t \), i.e.,

\[ \bar{N}(t) = N(\infty) - N(t) \quad . \]  

(2.18)
The expectation of $N(t)$ is given by

$$E[N(t)] = ae^{-bt}. \quad (2.19)$$

### 2.3.3 Conditional Distribution and Expectation of $N(t)$

If we have already observed $y$ faults, it is useful to know the distribution of the number of faults yet to be detected. In other words, the conditional distribution of $N(t)$, given that $N(t) = y$, is

$$P\{K(t) = x | N(t) = y\} = \frac{P\{N(t) = x, N(t) = y\}}{P\{N(t) = y\}}. \quad (2.20)$$

Now the event $N(t) = x$ denotes occurrences over the time interval $(t, \infty)$ while the event $N(t) = y$ denotes occurrences over the interval $(0, t)$, i.e., these two events represent non-overlapping time intervals. From a basic property of the NHPP process, such events are independent of each other, so that we have

$$P\{N(t) = x | N(t) = y\} = P\{N(t) = x\}, \quad x = 0, 1, 2, \ldots \quad (2.21)$$

or
\[ P(N(\infty) - N(t) = x | N(t) = y) = \frac{(m(\infty) - m(t))^x}{x!} e^{-(m(\infty) - m(t))}. \]

Or, substituting for \( m(\infty) \) and \( m(t) \) from Equation (2.10), we get

\[ P(N(\infty) - N(t) = x | N(t) = y) = \frac{(a - a(1 - e^{-bt}))^x}{x!} e^{-(a - a(1 - e^{-bt}))}. \]

This yields

\[ P(\bar{N}(t) = x | N(t) = y) = \frac{(ae^{-bt})^x}{x!} e^{-ae^{-bt}}. \quad (2.22) \]

Finally, the expected number of faults to be detected, given \( N(t) = y \), is

\[ E[\bar{N}(t) | N(t) = y] = ae^{-bt}. \quad (2.23) \]

This conditional distribution is important for deciding whether the software system under development can be released or not. The decision should be made based on the number of faults remaining in the software because this quantity plays an important role in software reliability assessment. Suppose that the decision-maker conducts an experiment and finds \( y \) software faults by time \( t \). Then, a decision might be to
Accept if $\bar{N}(t) \leq n_0$
and
Reject if $\bar{N}(t) > n_0$,

where $n_0$ is some specified number. For this decision rule, the probability that the software system is accepted for a given number of failures $y$ by time $t$ is

$$P\{\text{Accept}\} = P\{\bar{N}(t) \leq n_0 | N(t) = y\}$$

and, using Equation (2.22), becomes

$$P\{\text{Accept}\} = \sum_{i=0}^{n_0} P\{\bar{N}(t) = i | N(t) = y\} . \quad (2.24)$$

The conditional expectation of $\bar{N}(t)$, given $N(t)=y$, is given by

$$E[\bar{N}(t) | N(t)=y] = E[\bar{N}(t)]$$

or

$$E[\bar{N}(t) | N(t)=y] = ae^{-bt} . \quad (2.25)$$
Therefore, the expected number of faults remaining in the software system at time $t$, given that $y$ errors have been detected during the testing period $t$, is simply the expected number of faults to be detected during $[t, \infty]$. 
2.4 SOFTWARE RELIABILITY AND DISTRIBUTION OF TIME BETWEEN FAILURES

2.4.1 Software Reliability

Let a sequence of random variables \( \{X_i, i = 1, 2, \ldots\} \) denote a sequence of times between software failures associated with the \( N(t) \) process. Then \( X_i \) denotes the time between the \((i-1)\)st and the \(i\)th failures. We also define

\[
S_n = \sum_{i=1}^{n} X_i, \quad n = 1, 2, \ldots \quad (2.26)
\]

which represents the time to the \(n\)th failure. Let \( \phi_1(x) \) be the Cumulative Distribution Function (cdf) of \( X_1 \), i.e.,

\[
\phi_{X_1}(x) \equiv P(X_1 \leq x). \quad (2.27)
\]

Note that the event \( \{X_1 > x\} \) implies that there are no failures during \((0, x]\), i.e., the event \( \{N(x) = 0\} \). Then, using Equation (2.15), the reliability function associated with the first failure time is given by

\[
R_{X_1}(x) \equiv P(X_1 > x) = P(N(x) = 0)
\]

2-18
or

\[ R_{X_1}(x) = e^{-a(1 - e^{-bx})}. \]  

(2.28)

Now, the cdf of \( X_1 \) can be written as

\[ \Phi_{X_1}(x) = 1 - R_{X_1}(x) \]

or

\[ \Phi_{X_1}(x) = 1 - e^{-a(1 - e^{-bx})}. \]  

(2.29)

The Probability Density Function (pdf) is defined as

\[ \phi_{X_1}(x) = \frac{d}{dx} \Phi_{X_1}(x) \]

so that

\[ \phi_{X_1}(x) = abe^{-bx}e^{-a(1 - e^{-bx})}. \]  

(2.30)

Next, consider the conditional probability distribution, \( \phi_{X_2|X_1}(x|s) \), of \( \{X_2|X_1\} \). The event \( \{X_2 > x|X_1 = s\} \) implies \{no failures in \((s,s+x]\)\}. Then the conditional
reliability function of the second failure, given that the first failure occurs at time $s$, is given by

$$R_{X_2|X_1}(x|s) = P(X_2 > x|X_1 = s)$$

$$= P(\text{no failures in } (s, s+x])$$

$$= P(N(s+x) - N(x) = 0)$$

$$= e^{-[m(s+x) - m(s)]}$$

$$= e^{-a[e^{-bs} - e^{-b(s+x)}]} . \quad (2.31)$$

From Equation (2.31), we obtain

$$\phi_{X_2|X_1}(x|s) = 1 - R_{X_2|X_1}(x|s)$$

$$= 1 - e^{-a[e^{-bs} - e^{-b(s+x)}]} \quad (2.32)$$

and

$$\phi_{X_2|X_1}(x|s) = \frac{d}{dx} \phi_{X_2|X_1}(x|s)$$

$$= abe^{-b(s+x)}e^{-a[e^{-bs} - e^{-b(s+x)}]} . \quad (2.33)$$

2-20
Combining Equations (2.29) and (2.33), we get the joint density of \( x_1 \) and \( x_2 \) as

\[
\begin{align*}
\phi_{x_1,x_2}(x_1,x_2) &= \phi_{x_2|x_1}(x_2|x_1) \phi_{x_1}(x_1) \\
&= (a e^{-bx_1}) (a e^{-b(x_1+x_2)}) \\
&= a^2 b e^{-bx_1} e^{-b(x_1+x_2)} \\
&= a^2 b e^{-b(x_1+x_2)} e^{-a(1-e^{-b})}.
\end{align*}
\]

Making the transformation \( s_1 = x_1 \), \( s_2 = x_1 + x_2 \), the joint density of \( S_1 \) and \( S_2 \) is

\[
\begin{align*}
\phi_{S_1,S_2}(s_1,s_2) &= a^2 b e^{-bs_1} e^{-bs_2} e^{-a(1-e^{-b})}. \\
&= a^2 b e^{-bs_1} e^{-bs_2} e^{-a(1-e^{-b})}.
\end{align*}
\]

In general, it can be shown that the conditional reliability function of \( X_k \), given \( S_{k-1} = s \), is given by

\[
R_{X_k|S_{k-1}}(x|s) = e^{-a(e^{-bs} - e^{-b(s+x)})}. \\
\]
2.4.2 Conditional Distribution of $X_k | S_{k-1}$

The conditional cdf and pdf are obtained from Equation (2.36) by recalling that $R(x) = 1 - \phi(x)$ and $\phi(x) = \frac{d}{dx} \phi(x)$. Thus, we have

$$\Phi_{X_k | S_{k-1}}(x | s) = 1 - e^{-a(e^{-bs} - e^{-b(s+x)})}$$

(2.37)

and

$$\phi_{X_k | S_{k-1}}(x | s) = a e^{-b(s+x)} e^{-a(e^{-bs} - e^{-b(s+x)})}$$

(2.38)

respectively.

As can be seen from the above equations, the time to the next failure depends on the time when the last failure occurs. It should be noted that the distributions of times between failures are improper, i.e.,

$$\Phi_{X_k | S_{k-1}}(\infty | .) = 1 - e^{-ae^{-bs}} < 1 .$$

(2.39)

This is due to the fact that the event \{no failures in $(0,\infty)$\} is allowed in our model. Hence, the expectations of these quantities do not exist.

2-22
2.4.3 Joint Density of Waiting Times

As defined above, \( \{x_k, k = 1, 2, \ldots \} \) denotes the sequence of times between software failures. Then

\[
S_n = \sum_{i=1}^{n} x_i, \quad n = 1, 2, \ldots
\]

is called the waiting time to the \( n \)th software failure. This quantity is quite important for estimation of parameters \( a \) and \( b \) and, hence, we obtain the distribution of \( \{S_1, S_2, \ldots, S_n\} \). The distribution is obtained by using an approach similar to that used for getting Equation (2.34). The result is summarized in the following theorem.

**Theorem.** The joint probability density of \( S_1, S_2, \ldots, S_n \) is given by

\[
\phi_{S_1, \ldots, S_n}(s_1, \ldots, s_n) = (ab)^n \cdot e^{\sum_{i=1}^{n} -s_i} \cdot e^{-a(1-e^{s_i}) - bs_n} (2.40)
\]

where \( s_1, s_2, \ldots, s_n \) denote the realizations of \( S_1, S_2, \ldots, S_n \), respectively.

The density can also be written as

\[
\phi_{S_1, \ldots, S_n}(s_1, \ldots, s_n) = e^{-m(s_n)} \prod_{k=1}^{n} \lambda(s_k) (2.41)
\]

2-23
where \( \lambda(s_k) = \frac{d}{ds_k}\{m(s_k)\} \) and \( m(s_k) = a(1 - e^{-bs_k}) \). For a proof of this theorem, see [COX66] and [DON75].

Equation (2.40) will be used later to estimate \( a \) and \( b \) based on observed data \( s = (s_1, \ldots, s_n) \).

2.4.4 Joint Counting Probability

The property of independent increments, along with Equations (2.8) and (2.12), provides a complete statistical characterization for NHPP so that the joint counting probability can be determined for any collection of times \( 0 < t_1 < t_2 < \ldots < t_n \). That is, with \( t_0 = 0 \), \( Y_0 = 0 \).

\[
P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n)
= \prod_{i=1}^{n} P(N(t_i) - N(t_{i-1}) = y_i - y_{i-1})
= \prod_{i=1}^{n} \frac{[m(t_i) - m(t_{i-1})]^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} e^{-m(t_{n})}.
\]

Equation (2.42) is needed for estimating the parameters \( a \) and \( b \) for given data \( \{(y_i, t_i), i = 1, 2, \ldots, n\} \).
2.5 ESTIMATION OF MODEL PARAMETERS FROM FAILURE DATA

The basic models for the failure process and performance measures were developed in Sections 2.2 and 2.3, respectively. In order to use these models for software performance assessment, the only parameters to be specified are the total expected number of errors to be detected, $a$, and the error detection rate per error, $b$. In other words, for given $a$ and $b$, various useful quantities can be computed from the relevant equations in sections 2.2 and 2.3.

In general, $a$ and $b$ are not known for a specific software system and are estimated from the available data generated during testing. However, that is not the only way to get $a$ and $b$. One may be willing to extrapolate these values based on the data from one or more "similar" systems. Another method would be to use a Bayesian approach, whereby knowledge about $a$ and $b$ can be expressed as prior distributions and used for performance assessment. This approach can also be used in conjunction with available data and is specially useful when failure data are scarce or expensive to collect.

The purpose of this section is to describe methods for estimating $a$ and $b$ from failure data. Use of these
methods is illustrated later via failure data from operational systems. Such data are generally available as

(i) total number of failures in given time intervals; and/or as

(ii) times between failures.

Most of the available data is given in the form of number of failures in given time intervals; the data on times between failures is very rare. Nevertheless, both of these cases are considered below.

2.5.1 Estimation When Cumulative Failures Are Given

We first consider the case when data are available as cumulative number of failures in given time intervals. Suppose \( y_1, y_2, \ldots, y_n \) are the cumulative number of failures detected by times \( t_1, t_2, \ldots, t_n \), respectively. This can also be written as data pairs \( \{(y_i, t_i), i = 1, 2, \ldots, n\} \). Thus, the number of failures in time interval \( (t_{i-1}, t_i) \) is \( y_i - y_{i-1} \) for \( i = 1, 2, 3, \ldots, n \), where \( t_0 = 0 \) and \( y_0 = 0 \).

We will obtain the Maximum Likelihood Estimates \( \hat{a} \) and \( \hat{b} \) of \( a \) and \( b \), respectively. To do this, we first write the joint density and obtain the likelihood function, and then the log-likelihood function. Next, we take the partial derivatives of the log-likelihood function with respect to \( a \) and \( b \) and equate them to zero for maximization.

2-26
The solutions of the resulting two equations are the desired values \((\hat{a}, \hat{b})\).

Now, to get the joint density, we note that in our notations \(y_1, y_2, \ldots, y_n\) are the observed values of \(N(t_1), N(t_2), \ldots, N(t_n)\), respectively. Hence, from Equation (2.42),

\[
P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n)
= \prod_{i=1}^{n} \frac{[m(t_i) - m(t_{i-1})]^{y_i-y_{i-1}}}{(y_i-y_{i-1})!} e^{-(m(t_i) - m(t_{i-1}))}
\]

where \(m(t_i) = a(1-e^{-bt_i})\).

It is well known that the likelihood function for the parameters is simply the joint density of \(y_1, y_2, \ldots, y_n\), with these values considered as known constants. Substituting for \(m(t_i)\) in Equation (2.43), the likelihood function for \((a,b)\), given the data \((t,Y)\), is

\[
L(a,b|Y,t) = \prod_{i=1}^{n} \left\{ a(e^{-bt_1-1} - e^{-bt_i}) \right\}^{y_i-y_{i-1}} \frac{e^{-bt_i}}{(y_i-y_{i-1})!} e^{-a(1-e^{-bt_n})}.
\]

Taking the natural logarithm of Equation (2.44) yields:
\[ \ln L(a, b | y, t) = \sum_{i=1}^{n} (y_i - y_{i-1}) \ln a + \sum_{i=1}^{n} (y_i - y_{i-1}) \]
\[ \cdot \ln (e^{-bt_i - e^{-bt_i}}) - \sum_{i=1}^{n} \ln (y_i - y_{i-1})! - a(1 - e^{-bn}). \] (2.45)

As mentioned above, the maximum likelihood estimates (mle's) are those values of a and b which maximize \( \ln L(a, b | t, y) \), i.e., which satisfy (for brevity we write L to denote \( L(a, b | t, y) \))

\[ \frac{\partial \ln L}{\partial a} = 0 \]

and

\[ \frac{\partial \ln L}{\partial b} = 0. \] (2.46)

By taking the partial derivatives of Equation (2.45) and equating them to zero, we obtain, after some simplification (recall that \( y_0 = 0 \)),

\[ a(1 - e^{-bn}) = y_n, \] (2.47)

and
The solutions of the resulting two equations are the desired values \((a, b)\).

Now, to get the joint density, we note that in our notations \(y_1, y_2, \ldots, y_n\) are the observed values of \(N(t_1), N(t_2), \ldots, N(t_n)\), respectively. Hence, from Equation (2.42),

\[
P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n) = \frac{n \prod_{i=1}^{n} \left[ m(t_i) - m(t_{i-1}) \right]^{y_i - y_{i-1}} e^{-\left( m(t_i) - m(t_{i-1}) \right)}}{(y_i - y_{i-1})!}
\]

where \(m(t_i) = a(1 - e^{- bt_i})\).

It is well known that the likelihood function for the parameters is simply the joint density of \(y_1, y_2, \ldots, y_n\), with these values considered as known constants. Substituting for \(m(t_i)\) in Equation (2.43), the likelihood function for \((a, b)\), given the data \((t, y)\), is

\[
L(a, b | y, t) = \prod_{i=1}^{n} \frac{a(e^{-bt_{i-1}} - e^{-bt_i})^{y_i - y_{i-1}}}{(y_i - y_{i-1})!}
\]

Taking the natural logarithm of Equation (2.44) yields:

2-27
\[ \ln L(a, b | \mathbf{y}, t) = \sum_{i=1}^{n} (y_i - y_{i-1}) \ln a + \sum_{i=1}^{n} (y_i - y_{i-1}) \]

\[ -bt_{i-1} - bt_i - \ln(e^{bt_i - e^{bt_{i-1}}}) - \sum_{i=1}^{n} -bt_i \]

\[ \cdot \ln(y_i - y_{i-1})! - a(1 - e^{-bt_i}) \cdot n. \quad (2.45) \]

As mentioned above, the maximum likelihood estimates (MLE's) are those values of \(a\) and \(b\) which maximize \(\ln L(a, b | \mathbf{y}, t)\), i.e., which satisfy (for brevity we write \(L\) to denote \(L(a, b | \mathbf{y}, t)\))

\[ \frac{\partial \ln L}{\partial a} = 0 \]

and

\[ \frac{\partial \ln L}{\partial b} = 0. \quad (2.46) \]

By taking the partial derivatives of Equation (2.45) and equating them to zero, we obtain, after some simplification (recall that \(y_0 = 0\)),

\[ a(1 - e^{-bt_n}) = y_n, \quad (2.47) \]

and

2-28
As can be easily seen, all the quantities in Equations (2.47) and (2.48) are known except $a$ and $b$, which are to be estimated. These equations do not yield simple analytical forms and we use numerical methods for their solution. The resulting values of $a$ and $b$ are the mle's $\hat{a}$ and $\hat{b}$, respectively.

It should be pointed out that, even though the mle's are the desired values, it is often useful to study the log-likelihood surface as a function of parameters $a$ and $b$. For given data, a plot of the log-likelihood surface can be obtained by solving Equation (2.45) for a grid of values of $a$ and $b$. If the plot is flat, it would indicate a large variability associated with the mle's while a sharp surface is an indicator of low variability. A surface with sharp rises and falls might cause problems in numerical solution of Equations (2.47) and (2.48), while a well-behaved surface would ensure rapid convergence to the values $\hat{a}$, $\hat{b}$.
2.5.1.1 Confidence Region for \((a,b)\)

In addition to the mle's \(\hat{a}, \hat{b}\), we generally want to quantify the region in which the true values \(a, b\) might lie with a specified degree of confidence. This is referred to as obtaining the \(100(1-u)\%\) joint confidence region for \((a,b)\). In general, it is not possible to get the exact confidence region [FIN76] because the true distribution of \((\hat{a}, \hat{b})\) is unknown. However, mle's have a very desirable property that they are asymptotically normally distributed, if the sample size is large.

Also of great usefulness is the invariance property of the mle's, i.e., a function of \((a,b)\) can be estimated by using the mle's \(\hat{a}, \hat{b}\) and this function will also be a mle. This will be useful for estimating \(\bar{N}(t), R(t)\), etc.

Formally, as indicated above, the mle's are normally distributed for large \(n\), i.e.,

\[
\begin{pmatrix}
\hat{a} \\
\hat{b}
\end{pmatrix} \sim N \left( \begin{pmatrix} a \\ b \end{pmatrix}, \Sigma_{\text{cov}} \right) \quad \text{as } n \to \infty. \quad (2.49)
\]

The variance-covariance matrix represents
\[ \Sigma_{\text{cov}} = \begin{pmatrix} \text{Var}(a) & \text{Cov}(a,b) \\ \text{Cov}(b,a) & \text{Var}(b) \end{pmatrix} \]

and is given by

\[ \Sigma_{\text{cov}} = \begin{pmatrix} r_{aa} & r_{ab} \\ r_{ba} & r_{bb} \end{pmatrix}^{-1} \]

(2.50)

where

\[ r_{ij} = -E \frac{\partial^2 \ln L}{\partial i \partial j}, \quad i,j = a,b. \]

That is,

\[ r_{aa} = -E \frac{\partial^2 \ln L}{\partial a^2} \]

(2.51)

\[ r_{ab} = r_{ba} = -E \frac{\partial^2 \ln L}{\partial a \partial b} \]

(2.52)

\[ r_{bb} = -E \frac{\partial^2 \ln L}{\partial b^2} \]

(2.53)
Taking the appropriate partial derivatives of Equation (2.45) and substituting in Equations (2.51), (2.52), and (2.53), we obtain, after some simplification, (recall that \(E[N(t_i)] = m(t_i) = a(1-e^{-bt_i})\)):

\[
r_{aa} = \frac{1}{a} \sum_{i=1}^{n} (e^{-bt_i} - 1 - e^{-bt_i}), \quad (2.54)
\]

\[
r_{ab} = r_{ba} = t_n e^{-bt_n}, \quad (2.55)
\]

and

\[
r_{bb} = a \sum_{i=1}^{n} \frac{(t_i-1)^2 e^{-bt_i} - bt_i}{(e^{-bt_i} - 1 - e^{-bt_i})} - at_n^2 \cdot e^{-bt_n}. \quad (2.56)
\]

Substituting these expressions in Equation (2.50), we get the variance-covariance matrix for \((\hat{a}, \hat{b})\). Thus, the asymptotic distribution of \((\hat{a}, \hat{b})\) is completely specified if \((a, b)\) are known. However, in practice, \((a, b)\) are not known. Therefore, we use their estimates, \((\hat{a}, \hat{b})\), in Equations (2.49), (2.54), (2.55), and (2.56) to get estimates of the parameters of the asymptotic bivariate normal distribution.
Now, the correlation coefficient between \( \hat{a} \) and \( \hat{b} \) is estimated as

\[
\rho_{\hat{a}, \hat{b}} = \frac{\text{Cov}(\hat{a}, \hat{b})}{\sqrt{\text{Var}(\hat{a}), \text{Var}(\hat{b})}},
\]

(2.57)

where \( \text{Var}(\hat{a}), \text{Var}(\hat{b}), \text{Cov}(\hat{a}, \hat{b}) \) are obtained from Equations (2.50) to (2.53).

Finally, to obtain the 100(1-\( \alpha \))% confidence regions for \( a \) and \( b \), we use the following approximation ([ROU73])

\[
\ln L(\hat{a}, \hat{b} | y, t) - \ln L(a, b | y, t) = \frac{1}{2} \chi^2_{2, \alpha}
\]

or

\[
\ln L(a, b | y, t) = \ln L(\hat{a}, \hat{b} | y, t) - \frac{1}{2} \chi^2_{2, \alpha}
\]

(2.58)

where \( \ln L(\hat{a}, \hat{b} | y, t) \) represents the value of the log-likelihood function at \( a = \hat{a} \) and \( b = \hat{b} \).

Substituting Equation (2.45) in Equation (2.58), we get

2-33
\[
\sum_{i=1}^{n} (y_i - y_{i-1}) \ln a + \sum_{i=1}^{n} (y_i - y_{i-1}) \ln (e^{-bt_i} - e^{-bt_{i-1}}) \\
- \sum_{i=1}^{n} \ln((y_i - y_{i-1})!) - a(1 - e^{-bt_n}) = C, \quad (2.59)
\]

where

\[
C = \ln L(\hat{a}, \hat{b} | \mathbf{y}, t) - \frac{1}{2} \chi^2_{2; a}. \quad (2.60)
\]

Equation (2.59) defines a contour of the 100(1-\alpha)\% confidence region. For given data, \(\hat{a}, \hat{b},\) and \(\alpha,\) Equation (2.59) can be solved for those values of \(a\) and \(b\) which satisfy it. (For computational purposes, it is easier to take values of \(a (> \hat{a})\) and solve for the corresponding values of \(b.\))

2.5.2 Estimation When Times Between Failures Are Given

Now we consider the case when data is available in the form of times between individual failures. As mentioned earlier, such data is not common and is rarely available.

Recall that \(X_1, X_2, \ldots, X_n\) denote the times between \(n\) failures and \(S_n = \sum_{i=1}^{n} X_i.\) Then the data is in the form
\( x = (x_1, x_2, \ldots, x_n) \) and \( s_n = \sum_{i=1}^{n} x_i \). The distribution of times between failures was discussed in section 2.4.3 and is obtained from Equations (2.40) and (2.41), as

\[
\phi_{S_1, \ldots, S_n}(s_1, \ldots, s_n) = (ab)^n e^{-bs} \sum_{i=1}^{n} e^{-a(1-e^{-n})} .
\]

The likelihood function for \( a, b \), given \( s \), is the same as above and can be written as

\[
L(a, b | s) = (ab)^n \cdot e^{-bs} \sum_{i=1}^{n} e^{-a(1-e^{-n})} . \tag{2.61}
\]

Then the log (natural) likelihood is

\[
\ln L(a, b | s) = n \ln a + n \ln b - bs \sum_{i=1}^{n} s_i - a(1-e^{-n}) . \tag{2.62}
\]

To get the maximum likelihood estimates \( \hat{a}, \hat{b} \), we take the partial derivatives of Equation (2.62) and equate them to zero, i.e.,

\[
\frac{\partial \ln L}{\partial a} = 0 , \tag{2.63}
\]
and

\[ \frac{\partial \ln L}{\partial b} = 0. \]  \hspace{1cm} (2.64)

These equations yield

\[ \frac{n}{a} = 1 - e^{-bs_n} \]  \hspace{1cm} (2.65)

and

\[ \frac{n}{b} = a s_n \cdot e^{-bs_n} + \sum_{i=1}^{n} s_i. \]  \hspace{1cm} (2.66)

As in the first case, these equations do not yield simple analytical solutions and have to be solved numerically. The solutions of Equations (2.65), and (2.66) are the mle's \( \hat{a} \) and \( \hat{b} \).

Regarding the asymptotic distribution of \((\hat{a}, \hat{b})\), recall that (see section 2.4.2) the joint density of \(S_1\), \(\ldots, S_n\) is improper. Therefore, the asymptotic properties of mle's do not hold in this case.

To obtain the \(100(1-\alpha)\%\) confidence regions for \((a, b)\), we use the same approximation as was used in section 2.5.1, viz.
\[ \ln L(\hat{a}, \hat{b} \mid \mathbf{s}) - \ln L(a, b \mid \mathbf{s}) = \frac{1}{2} \chi_{2; \alpha}^2. \]  

(2.67)

From Equations (2.62) and (2.67), a contour of the

100(1-\alpha)\% confidence region is obtained as

\[ n \hat{\ln a} + n \hat{\ln b} - b \sum_{i=1}^{n} s_i - a(1 - e^{-b s}) = C, \]  

(2.68)

where

\[ C = \ln L(\hat{a}, \hat{b} \mid \mathbf{s}) - \frac{1}{2} \chi_{2; \alpha}^2. \]  

(2.69)

As before, Equation (2.68) can be solved for given

\( s, \hat{a}, \hat{b}, \) and \( \alpha \) to get the desired contours.
2.6 GOODNESS-OF-FIT TEST

In this section, we describe the Kolmogorov-Smirnov goodness-of-fit test (K-S Test) to check whether the NHPP model developed in sections 2.2 and 2.5 provides a good fit to a given set of failure data.

Consider the case when the data are given as a sequence of software failure times \( s = (s_1, s_2, \ldots, s_n) \). We want to test whether the events \( s \) are generated from a NHPP. Suppose that \( 0 < S_1 \leq S_2 \leq \ldots \leq S_n \) are the random times at which the first \( n \) events occur in a NHPP with unknown mean value function \( m(t) \). We wish to test the simple hypothesis

\[
H_0: m(t) = m_0(t) \text{ for } t \geq 0,
\]

versus

\[
H_1: m(t) \neq m_0(t) \text{ for } t \geq 0.
\]

Writing \( m_0(t) = a_0(1-e^{-b_0 t}) \), the hypothesis \( H_0 \) can be written as

\[
\text{2-38}
\]
\[
H_0: \quad m(t) = a_0 (1 - e^{-b_0 t}) \quad \text{for } t \geq 0. \quad (2.70)
\]

For testing purposes, we need the joint conditional distribution of the failure times. The following theorem is useful in deriving this distribution.

**Theorem.** Given that \( N(t) = n \), the \( n \) failure times \( 0 < S_1 < S_2 < \ldots < S_n \) in the interval \([0,t]\) are random variables whose joint conditional distribution is the same as the distribution of the order statistics of a random sample of size \( n \) from the distribution \( G(x) = \frac{m(x)}{m(t)} \) for \( 0 < x < t \).

For proof of this Theorem, see Cox and Lewis [COX66].

**Corollary.** Given that \( S_n = t \), the \((n-1)\) failure times \( 0 < S_1 < S_2 < \ldots < S_{n-1} \) have the same joint conditional distribution as the order statistics of a random sample of size \((n-1)\) from the distribution \( G(x) = \frac{m(x)}{m(t)} \).

This Corollary easily follows from the above Theorem. Using this Corollary, we reduce the hypothesis of Equation (2.70) to

\[
H_0: \quad G(x) = G_0(x) = \frac{m_0(x)}{m_0(t)} \quad \text{for } 0 \leq x \leq t. \quad (2.71)
\]
For our case we have

\[ H_0: G(x) = \frac{1-e^{-b_0x}}{1-e^{-b_0t}} \text{ for } 0 \leq x \leq t. \quad (2.72) \]

Note that the expression in Equation (2.72) represents a truncated exponential distribution.

We now consider the Kolmogorov-Smirnov (K-S) goodness-of-fit test [ROM76, ROV73]. Given the values of a random sample of size \( n-1, s_1, s_2, \ldots, s_{n-1} \), we define the sample cdf by \( H_{n-1}(x) = k/(n-1) \), where \( k \) is the number of sample values \( \leq x \). Thus, \( H_{n-1}(x) \) is a step function which is zero for \( x \) less than \( s_1 \), has a jump of \( 1/(n-1) \) at each \( s_k \), and is 1 for \( x \) greater than or equal to \( s_{n-1} \). That is,

\[
H_{n-1}(x) = \begin{cases} 
0 & , \ x < s_1 \\
\frac{k}{(n-1)}, & s_{k-1} < x < s_k, \ k=2,3,\ldots,n-1. \\
1 & , \ x \geq s_{n-1}
\end{cases} \quad (2.73)
\]

Since \( H_{n-1} \) is a step function and \( G \) is monotonically increasing and continuous, it suffices to test the absolute
deviations at the sample points $s_k$, $k = 1, 2, \ldots, n-1$, and then take the maximum of these $(n-1)$ values. The following procedure is used for calculating the test statistic $D$. For each $k = 1, 2, \ldots, n-1$, set

$$D_k = \max\{|G_0(s_k) - \frac{k}{n-1}|, |G_0(s_k) - \frac{k-1}{n-1}|\}.$$ 

Then set

$$D = \max_k \{D_k\} \quad (2.74)$$

If the value of $D$ calculated in Equation (2.74) is greater than or equal to the critical value $D_{n-1; \alpha}$, we reject the null hypothesis $H_0$ that $S_1, S_2, \ldots, S_{n-1}$ follow $G_0(x)$; otherwise we do not reject the null hypothesis. The critical values $D_{n-1; \alpha}$ associated with the K-S test at a level of significance $\alpha$ are available from statistical tables [ROH76, p. 661].

It should be noted that, if the parameters of $G_0(x)$ are estimated from the sample, the K-S test can be used but will give extremely conservative results. To achieve better results, the level of significance needs to be adjusted. One approach suggested by Allen [ALL78] is
to test at the 5% level of significance and use the critical value for the 20% level or test at the 1% level and use the critical value for 10% level. We will use this approach in our analyses in later sections.

Another use of the K-S test in our context is in developing confidence limits for the true cdf $G(x)$. For example, if we take a random sample of size $(n-1)$ and use it to construct the sample cdf $H_{n-1}(x)$, then we can be $100(1-\alpha)\%$ confident that the true cdf $G(x)$ does not deviate from $H_{n-1}(x)$ by more than $D_{n-1;\alpha}$. That is, the $100(1-\alpha)\%$ confidence limits for $G(x)$ are given by

$$H_{n-1}(x) - D_{n-1;\alpha} < G(x) < H_{n-1}(x) + D_{n-1;\alpha} \quad (2.75)$$

These limits are especially useful in the case when the parameters of $G_0(x)$ are to be estimated from the data. For this case, the null hypothesis $H_0$ will be rejected at a level of significance $\alpha$ if one or more points of $G_0(x)$ fall outside the $100(1-\alpha)\%$ confidence limits given by Equation (2.75). Otherwise, it will not be rejected.
2.7 ANALYSIS OF FAILURE DATA FROM NAVAL TACTICAL DATA SYSTEM (NTDS)

Jelinski and Moranda [JEL 72] first analyzed some software failure data from the U.S. Navy Fleet Computer Programming Center. Since then, this data set has been used by several investigators for model validation purposes. In this section, we analyze the same data set to see how good the NHPP model is in modelling these failures.

The data set was extracted from information about errors in the development of software for the real-time, multi-computer complex which forms the core of the Naval Tactical Data System (NTDS). The NTDS software consisted of some 38 different project schedules. Each module was supposed to follow three stages: the production (or development) phase, the test phase, and the user phase. Many of the "trouble reports" or "software anomaly reports" were generated whenever a system-level symptom of a deficiency was noted by operators or users. A proper trace back to the exact cause in software of this symptom was done by personnel familiar with the entire system. However, Jelinski and Moranda felt that it was better to analyze the data from isolated modules than from the total system, due to the fact that many
of the modules did not evolve in the fashion indicated. One of the larger modules, denoted by A-module, had the desired pattern. The times (in days) between failures for this module are shown in Table 2.1. Twenty-six software faults were found during the production phase and five additional faults during the test phase. The last fault was found on 4 Jan 1971. One fault was observed during the user phase on 20 Sept 1971 and two more faults (4 Oct 1971, 10 Nov 1971) during the test phase. This indicates that a re-work of the module had taken place after the user error was found. A more detailed description of the NTDS software can be found in [JEL72].

Data Analyses

The data in this case is available as times between software failures and hence the method described in section 2.5.2 will be used for estimation of parameters. We consider the first 26 data points of Table 2.1, for which \( n = 26 \) and \( s_{26}^2 = \sum_{k=1}^{26} x_k = 250 \) days.

To get an appreciation of the likelihood function associated with this data set, the log-likelihood from Equation (2.62) is plotted in Figure 2.2. We see that
### TABLE 2.1
SOFTWARE FAILURE DATA FROM NTDS

<table>
<thead>
<tr>
<th>ERROR NO.</th>
<th>TIME BETWEEN FAILURES $x_k, \text{ days}$</th>
<th>CUMULATIVE TIME $s_n = \sum x_k, \text{ days}$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Production (Checkout) Phase</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>2</td>
<td>12</td>
<td>21</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>32</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
<td>43</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>45</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>58</td>
</tr>
<tr>
<td>9</td>
<td>5</td>
<td>63</td>
</tr>
<tr>
<td>10</td>
<td>7</td>
<td>70</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>71</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
<td>77</td>
</tr>
<tr>
<td>13</td>
<td>1</td>
<td>78</td>
</tr>
<tr>
<td>14</td>
<td>9</td>
<td>87</td>
</tr>
<tr>
<td>15</td>
<td>4</td>
<td>91</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>92</td>
</tr>
<tr>
<td>17</td>
<td>3</td>
<td>95</td>
</tr>
<tr>
<td>18</td>
<td>3</td>
<td>98</td>
</tr>
<tr>
<td>19</td>
<td>6</td>
<td>104</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>105</td>
</tr>
<tr>
<td>21</td>
<td>11</td>
<td>116</td>
</tr>
<tr>
<td>22</td>
<td>33</td>
<td>149</td>
</tr>
<tr>
<td>23</td>
<td>7</td>
<td>156</td>
</tr>
<tr>
<td>24</td>
<td>91</td>
<td>247</td>
</tr>
<tr>
<td>25</td>
<td>2</td>
<td>249</td>
</tr>
<tr>
<td>26</td>
<td>1</td>
<td>250</td>
</tr>
<tr>
<td><strong>Test Phase</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>87</td>
<td>337</td>
</tr>
<tr>
<td>28</td>
<td>47</td>
<td>384</td>
</tr>
<tr>
<td>29</td>
<td>12</td>
<td>396</td>
</tr>
<tr>
<td>30</td>
<td>9</td>
<td>405</td>
</tr>
<tr>
<td>31</td>
<td>135</td>
<td>540</td>
</tr>
<tr>
<td><strong>User Phase</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>258</td>
<td>798</td>
</tr>
<tr>
<td><strong>Test Phase</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>16</td>
<td>814</td>
</tr>
<tr>
<td>34</td>
<td>35</td>
<td>849</td>
</tr>
</tbody>
</table>

2-45
the surface rises sharply along the b-axis and is relatively flat along the a-axis.

The maximum of this surface is obtained by solving Equations (2.65) and (2.66). Substituting the appropriate values from Table 2.6 in Equations (2.65) and (2.66) we get

\[
\frac{26}{a} = 1 - e^{-b(250)} \quad (2.76)
\]

and

\[
\frac{26}{b} = a(250) \cdot e^{-b(250)} + 250. \quad (2.77)
\]

Solving Equations (2.76) and (2.77) numerically, we get

\[\hat{a} = 33.99\]

and

\[\hat{b} = 0.00579\]

as the mle's for \(a\) and \(b\), respectively. The fitted mean value function is

2-47
\( \hat{m}(t) = 33.99(1 - e^{-0.00579t}) \). \hspace{1cm} (2.78)

and is shown in Figure 2.3, along with the actual data (determination of the confidence bounds will be discussed later).

Goodness-of-fit Test

We now perform the Kolmogorov-Smirnov goodness-of-fit test to check the adequacy of the fitted model. Now, using the Corollary and the results in Section 2.6, we conduct the test based on \((26 - 1) = 25\) points. The hypothesis, from Equation (2.71), is

\[
H_0: \ G_0(x) = \frac{e^{-b_0x}}{1-e^{-b_0(250)}} \text{ for } 0 < x < 250, \hspace{1cm} (2.79)
\]

and the sample cdf is

\[
H(x) = \begin{cases} 
0 & , \ x < s_1 \\
\frac{k}{25} & , \ s_{k-1} < x < s_k, \ k=2,3,\ldots,25 \\
1 & , \ x \geq s_{25} 
\end{cases} \hspace{1cm} (2.80)
\]
Figure 2.3. Plots of Mean Value Function and 90% Confidence Bounds for the N(t) Process (NTDS Data)
The values of $s_k$ and $H(s_k)$ are given in Table 2.2. To compute $G_0(s_k)$ for various $s_k$ values, we place $b_0$ by $b$ in Equation (2.79) and obtain Column 4 of Table 2.2. Entries in Columns 5 and 6 are easily obtained from Columns 3 and 4. Now, from Equations (2.47) and (2.79),

\[
D = \max_k \{ |G_0(s_k) - H(s_k)|, |G_0(s_k) - H(s_{k-1})| \}.
\]

In other words, $D$ is the largest entry in Columns 5 and 6 and is seen to be

\[ D = 0.2044. \]

To test at $\alpha = .05$, we use a critical value corresponding to $\alpha = .20$ as discussed in section 2.6.

From statistical tables,

\[ D_{25;0.2} = 0.208. \]

Since $D < D_{25;0.2}$, we accept the null hypothesis, $H_0$, at 5% level of significance.

The $100(1-\alpha)$% confidence limits for $G(x)$ can now be calculated from Equation (2.75). For example, for

2-50
TABLE 2.2
KOLOMOGOROV-SMIRNOV TEST
FOR THE NTDS DATA SET

| k | s_k | H(s_k) | G_0(s_k) | |G_0(s_k)-H(s_k)| | |G_0(s_k)-H(s_{k-1})|
|---|-----|--------|---------|-----------------|-----------------|------|
| 1 | 9 | 0.04 | 0.0664 | 0.0264 | 0.0664 |
| 2 | 21 | 0.08 | 0.1497 | 0.0697 | 0.1097 |
| 3 | 32 | 0.12 | 0.2211 | 0.1011 | 0.1411 |
| 4 | 36 | 0.16 | 0.2460 | 0.0860 | 0.1260 |
| 5 | 43 | 0.20 | 0.2882 | 0.0882 | 0.1282 |
| 6 | 45 | 0.24 | 0.2999 | 0.0599 | 0.0999 |
| 7 | 50 | 0.28 | 0.3286 | 0.0486 | 0.0886 |
| 8 | 58 | 0.32 | 0.3730 | 0.0530 | 0.0930 |
| 9 | 63 | 0.36 | 0.3996 | 0.0396 | 0.0796 |
| 10 | 70 | 0.40 | 0.4357 | 0.0357 | 0.0757 |
| 11 | 71 | 0.44 | 0.4407 | 0.0007 | 0.0407 |
| 12 | 77 | 0.48 | 0.4703 | 0.0097 | 0.0303 |
| 13 | 78 | 0.52 | 0.4751 | 0.0449 | 0.0049 |
| 14 | 87 | 0.56 | 0.5174 | 0.0426 | 0.0026 |
| 15 | 91 | 0.60 | 0.5355 | 0.0645 | 0.0245 |
| 16 | 92 | 0.64 | 0.5399 | 0.1001 | 0.0601 |
| 17 | 95 | 0.68 | 0.5532 | 0.1268 | 0.0868 |
| 18 | 98 | 0.72 | 0.5661 | 0.1539 | 0.1139 |
| 19 | 104 | 0.76 | 0.5915 | 0.1685 | 0.1285 |
| 20 | 105 | 0.80 | 0.5956 | 0.2044 | 0.1644 |
| 21 | 116 | 0.84 | 0.6395 | 0.2005 | 0.1605 |
| 22 | 149 | 0.88 | 0.7557 | 0.1243 | 0.0843 |
| 23 | 156 | 0.92 | 0.7776 | 0.1424 | 0.1024 |
| 24 | 247 | 0.96 | 0.9946 | 0.0346 | 0.0746 |
| 25 | 249 | 1.00 | 0.9982 | 0.0018 | 0.0382 |

2-51
\[ a = 0.05, \text{ we have } D_{25;0.05} = 0.264, \text{ so that the lower and upper confidence bounds are} \]

\[ L(x) = \max\{H(x) - 0.264, 0\} \]

and

\[ U(x) = \min\{H(x) + 0.264, 1\}, \]

where \( H(x) \) is given by Equation (2.80). The 95% bounds for \( G(x) \), along with \( G_0(x) \), are shown in Figure 2.4. We see that the fitted model seems to be adequate.

Having established that the model provides a good fit, various performance measures of interest can be obtained by substituting the estimated values of \( a \) and \( b \) in the appropriate equations of sections 2.3 and 2.4.

The estimated mean value function, as given in Equation (2.78), is \( \hat{m}(t) = 33.99(1-e^{-0.00579t}) \). A plot of \( \hat{m}(t) \) and the actual number of faults detected during the production period for this case was given in Figure 2.3. Also shown were the 90% confidence bounds for the \( N(t) \) process as computed from Equation (2.15).
Figure 2.4 95% confidence bounds for the conditional c.d.f. $G(x)$ and the fitted C.D.F. curve (NTDS data)
The 100(1-\alpha)\% confidence regions for a and b are obtained from Equations (2.68) and (2.69) following a procedure similar to the one detailed in section 2.7. These are shown in Figure 2.5 for \(a = 0.05, 0.25,\) and 0.50.

Finally, software reliability, \(R_{X_{27}|S_{26}}(x|250)\), can be computed from Equation (2.36). For example, the reliability values after \(x = 5, 10, 20,\) and 30 days are 0.796, 0.638, 0.417, and 0.280, respectively. Thus, the probability that the system will operate without any failures for 30 additional days is 0.28. As seen from the data in Table 2.1, the system did operate without any failures for 87 days subsequent to failure number 26.
Figure 2.8. Joint Confidence Regions for $a$ and $b$ (NTDS Data)
2.8 ANALYSIS OF FAILURE DATA FROM A LARGE SCALE SOFTWARE SYSTEM

The data to be analyzed in this section have been taken from a large scale project reported in Thayer et al. [THA76]. This project represents an initial delivery of a large command and control software package written in JOVIAL/J4 (JOVIAL is a higher order language generally used for Air Force Command and Control applications). It consists of 115,346 total source statements and 249 routines. Some other characteristics of this project are summarized in Table 2.3. The software was developed functionally, i.e., the project was divided into work units responsible for different functions. Software testing started with developing testing by the development personnel to demonstrate specific functional capabilities, test data extremes, etc.

2.8.1 Failure Data

The failure data used for this study is taken from the Software Problem Reports (SPR's) generated during the formal testing phases of this project. Formal testing, which comprises of validation and acceptance testing, began after development testing. Validation testing was
**TABLE 2.3**

**SOFTWARE PROJECT CHARACTERISTICS**

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size (Total source statement)</td>
<td>115,346</td>
</tr>
<tr>
<td>Number of routines</td>
<td>249</td>
</tr>
<tr>
<td>Language</td>
<td>JOVIAL/J4</td>
</tr>
<tr>
<td>Formal Requirements</td>
<td>To function level</td>
</tr>
<tr>
<td>Co-contractor</td>
<td>Yes</td>
</tr>
<tr>
<td>Subcontractor</td>
<td>No</td>
</tr>
<tr>
<td>Operating Mode</td>
<td>Batch</td>
</tr>
<tr>
<td>Formal Testing</td>
<td>24 Weeks</td>
</tr>
<tr>
<td>Validation</td>
<td>10</td>
</tr>
<tr>
<td>Acceptance</td>
<td>2</td>
</tr>
<tr>
<td>Integration</td>
<td>10</td>
</tr>
<tr>
<td>Operational Demonstration</td>
<td>2</td>
</tr>
</tbody>
</table>
performed by an independent test group at the subsystem level and demonstrated the approved software performance and requirements. Acceptance testing ran a subset of the Validation tests to demonstrate specific requirements. After Acceptance testing, the software underwent final Integration testing by an independent group. Integration testing demonstrated that the applications software correctly interfaced with the operating system and system support software. Finally, Operational Demonstration testing was done to demonstrate the software in an operational environment using an operational timeline and operational data. The data for this error data set was obtained from the four formal test phases (Validation, Acceptance, Integration, and Operational Demonstration) of the applications software. This is so because the majority of the errors analyzed were detected during formal testing.

The time period for the various phases of testing is validation (Jun 1-Aug 12), Acceptance (Aug 13-Aug 24), Integration (Aug 25-Oct 26), and Operational Demonstration (Oct 27-Nov 12) testing. In addition to the above data, operational data spanning a period of approximately nine months was also available and is used for comparison with the predicted values. The only time frame
readily available from the data was the calendar day. The data also contain the mistakes by the operators and the "explanatory" errors, i.e., corrections to make a change to a comment statement or those errors for which a "fix" is not to a routine. These explanatory errors do or do not indicate the type of change. Therefore, the original data was restructured into four sets of data denoted by DS1, DS2, DS3, and DS4 [SUK76]. The description and the total number of faults detected during the formal testing phases for each data set are given in Table 2.4.

In this analysis, the number of software faults detected during formal testing is counted on a weekly basis. Also, for each data set, the software faults detected during the first nine weeks are eliminated due to the fact that we are interested in analyzing the software failures over the period when they are decreasing. The number of SPR's for the 15-week period for the four cases (DS1 to DS4) are given in Table 2.5.

2.8.2 Estimation of Parameters

As seen in Table 2.5, the data for this project are in the form \((t_1, y_1), (t_2, y_2), \ldots, (t_{15}, y_{15})\), i.e.,
TABLE 2.4

DESCRIPTION OF THE DATA SETS

<table>
<thead>
<tr>
<th>DATA SET</th>
<th>DESCRIPTION</th>
<th>TOTAL NUMBER OF FAILURES</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Formal Testing (24 weeks)</td>
</tr>
<tr>
<td>DS1</td>
<td>Original Data - TT - EX1 - EX2</td>
<td>2191</td>
</tr>
<tr>
<td>DS2</td>
<td>Original Data - TT - EX1</td>
<td>2621</td>
</tr>
<tr>
<td>DS3</td>
<td>Original Data - TT</td>
<td>4367</td>
</tr>
<tr>
<td>DS4</td>
<td>Original Data - TT - EX2</td>
<td>3937</td>
</tr>
</tbody>
</table>

TT represents the mistakes by the operators.

EX1 represents the explanatory errors which do not indicate what type of change (module, documentation, compool, data base) was involved.

EX2 represents the explanatory errors which indicate type of change.
<table>
<thead>
<tr>
<th>WEEK</th>
<th>DS1</th>
<th>DS2</th>
<th>DS3</th>
<th>DS4</th>
<th>DS1</th>
<th>DS2</th>
<th>DS3</th>
<th>DS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>203</td>
<td>238</td>
<td>369</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
<td>334</td>
</tr>
<tr>
<td>2</td>
<td>136</td>
<td>179</td>
<td>292</td>
<td>667</td>
<td>255</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>3</td>
<td>183</td>
<td>222</td>
<td>398</td>
<td>1065</td>
<td>1065</td>
<td>1065</td>
<td>1065</td>
<td>1065</td>
</tr>
<tr>
<td>4</td>
<td>47</td>
<td>569</td>
<td>712</td>
<td>1180</td>
<td>1180</td>
<td>1180</td>
<td>1180</td>
<td>1180</td>
</tr>
<tr>
<td>5</td>
<td>46</td>
<td>615</td>
<td>73</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>6</td>
<td>71</td>
<td>686</td>
<td>768</td>
<td>1263</td>
<td>1263</td>
<td>1263</td>
<td>1263</td>
<td>1263</td>
</tr>
<tr>
<td>7</td>
<td>54</td>
<td>740</td>
<td>856</td>
<td>1413</td>
<td>1413</td>
<td>1413</td>
<td>1413</td>
<td>1413</td>
</tr>
<tr>
<td>8</td>
<td>57</td>
<td>797</td>
<td>95</td>
<td>1555</td>
<td>1555</td>
<td>1555</td>
<td>1555</td>
<td>1555</td>
</tr>
<tr>
<td>9</td>
<td>80</td>
<td>877</td>
<td>934</td>
<td>1727</td>
<td>1727</td>
<td>1727</td>
<td>1727</td>
<td>1727</td>
</tr>
<tr>
<td>10</td>
<td>64</td>
<td>941</td>
<td>1026</td>
<td>1897</td>
<td>1897</td>
<td>1897</td>
<td>1897</td>
<td>1897</td>
</tr>
<tr>
<td>11</td>
<td>27</td>
<td>968</td>
<td>1130</td>
<td>2097</td>
<td>2097</td>
<td>2097</td>
<td>2097</td>
<td>2097</td>
</tr>
<tr>
<td>12</td>
<td>64</td>
<td>1010</td>
<td>1215</td>
<td>2186</td>
<td>2186</td>
<td>2186</td>
<td>2186</td>
<td>2186</td>
</tr>
<tr>
<td>13</td>
<td>59</td>
<td>1065</td>
<td>1266</td>
<td>2273</td>
<td>2273</td>
<td>2273</td>
<td>2273</td>
<td>2273</td>
</tr>
<tr>
<td>14</td>
<td>62</td>
<td>1127</td>
<td>1372</td>
<td>2384</td>
<td>2384</td>
<td>2384</td>
<td>2384</td>
<td>2384</td>
</tr>
<tr>
<td>15</td>
<td>11</td>
<td>1138</td>
<td>1456</td>
<td>253</td>
<td>253</td>
<td>253</td>
<td>253</td>
<td>253</td>
</tr>
</tbody>
</table>

**TABLE 2.5**

SOFTWARE DATA SETS DS1 TO DS4

2-61
as the number of failures in specified time intervals. Hence, the estimates \( \hat{a} \) and \( \hat{b} \) are obtained by simultaneously solving Equations (2.47) and (2.48). Thus, by substituting the data set DS1 in Equations (2.47) and (2.48) and solving, we get

\[
\hat{a} = 1348, \quad \hat{b} = 0.124,
\]

and the fitted mean value function is

\[
\hat{m}(t) = 1348(1 - e^{-0.124t}), \quad t \geq 0.
\]

This is also an estimate of the expected number of software failures observed by time \( t \). A plot of the actual cumulative number of failures and the fitted values is given in Figure 2.6.

2.8.3 **Goodness-of-fit Test**

The goodness-of-fit test is now conducted following the procedure discussed in section 2.6. Since the sample size is 15, the null hypothesis to be tested can be written as
Figure 2.6. Actual and Expected Cumulative Number of Failures and 90% confidence bounds for the N(t) process for data set DS1.
H₀: \( G₀(tᵢ) = \frac{-b₀tᵢ}{1-e^{-b₀(tᵢ)}} \) for \( i=1,2,...,15 \), (2.81)

and the sample cdf as

\[
H(x) = \begin{cases} 
0 & , x < t₁ \\
y₁/y₁₅ & , tᵢ₋₁ < x < tᵢ , \ i=2,3,...,15 \end{cases} \quad (2.82)
\]

The computed values of \( H(x) \) for various \( tᵢ \) are given in column 2 of Table 2.6.

Now we substitute \( b₀ = \hat{b} = 0.124 \) in Equation (2.81) and compute the value of \( G₀(tᵢ) \) for \( i=1,2,...,15 \). These values are given in column 3 of Table 2.6. Columns 4 and 5 of this table are the quantities needed to find \( D = \max\{D_k\} \) (see Equation (2.74)). From these columns we find the value of \( D \) to be 0.096 corresponding to \( tᵢ = 9 \).

To find the critical value corresponding to sample size 15 and \( \alpha = .05 \), we first note that the parameters had to be estimated in this case. As mentioned in section 2.6, for a situation like this, a suggested approach is to take \( \alpha = .20 \) to get good results. From
| $t_i$ | $H(t_i)$ | $G_0(t_i)$ | $|G_0(t_i) - H(t_i)|$ | $|G_0(t_i) - H(t_{i-1})|$ |
|------|---------|-----------|-----------------|------------------|
| 1    | 0.1784  | 0.1381    | 0.0403          | 0.1381           |
| 2    | 0.2979  | 0.2601    | 0.0378          | 0.0817           |
| 3    | 0.4587  | 0.3679    | 0.0908          | 0.0700           |
| 4    | 0.5000  | 0.4631    | 0.0369          | 0.0044           |
| 5    | 0.5404  | 0.5472    | 0.0068          | 0.0472           |
| 6    | 0.6028  | 0.6215    | 0.0187          | 0.0811           |
| 7    | 0.6503  | 0.6872    | 0.0369          | 0.0844           |
| 8    | 0.7004  | 0.7452    | 0.0448          | 0.0949           |
| 9    | 0.7707  | 0.7964    | 0.0257          | 0.096            |
| 10   | 0.8269  | 0.8416    | 0.0147          | 0.0709           |
| 11   | 0.8506  | 0.8816    | 0.031           | 0.0547           |
| 12   | 0.8875  | 0.9169    | 0.0294          | 0.0663           |
| 13   | 0.9359  | 0.9481    | 0.0122          | 0.0606           |
| 14   | 0.9903  | 0.9757    | 0.0146          | 0.0398           |
| 15   | 1.0000  | 1.0000    | 0.0000          | 0.0097           |
The observed value $D = 0.096$ is less than the critical value $0.266$ and hence we accept the null hypotheses of Equation (2.76). Thus we conclude that at 5% level of significance the model

$$P(N(t) = y) = \frac{1348(1-e^{-0.124t})}{y!}(e^{-1348(1-e^{-0.124t})})$$

can be considered to provide an adequate fit to data set DS1.

To further check the adequacy of fit, we compute 95% confidence bounds on $G(t_i)$. From Equation (2.75), these bounds are given by

$$H(t_i) - D_{15;0.05} < G(t_i) < H(t_i) + D_{15;0.05}.$$ 

From the statistical tables, $D_{15;0.05} = 0.366$ and hence the 95% confidence bounds are given by $H(t_i) \pm 0.366$. A plot of these bounds and the fitted values are shown in Figure 2.7.
Figure 2.7. 95% confidence bounds for the conditional c.d.f. $G(t_1)$ and the fitted curve for DS1 data.
2.8.4 Confidence Regions for \((a,b)\)

To get an appreciation of the variability in the estimated values of \(a\) and \(b\), we now construct confidence regions for \((a,b)\). Such regions are given by Equations (2.59) and (2.60). For \(\alpha = .05\), the 95% joint confidence region will be the solution of the following equation:

\[
\ln \mathcal{L}(\hat{a}, \hat{b} | \mathcal{Y}, \mathcal{T}) = \ln \mathcal{L}(a, b | \mathcal{Y}, t) - \frac{1}{2} \chi^2_{2, .05},
\]

where

\[
\ln \mathcal{L}(a, b | \mathcal{Y}, t) = \sum_{i=1}^{15} (y_i - y_{i-1}) \ln(1348) + \sum_{i=1}^{15} (y_i - y_{i-1})
- .124 t_{i-1} - .124 t_i
- .124 t_{15}
- .124 t_i
- 1348 (1-e^{.124 t_{15}}).
\]

Data \((y_1, t_1), (y_2, t_2), \ldots, (y_{15}, t_{15})\) were given in Table 2.5 and

\[
\chi^2_{2, .05} = 0.103.
\]
A plot of this region is shown in Figure 2.8. From this plot we see that, even though the most likely values of \( a \) and \( b \), based on the data, are \( \hat{a} = 1348, \hat{b} = 0.124 \), the true values can vary over the entire region contained in the 95% contour. Values \( a = 1450, b = 0.11 \) will be acceptable (with 95% confidence) and so will \( a = 1250, b = 0.14 \). 50% and 75% confidence regions are also shown in Figure 2.8 and can be similarly interpreted.

2.8.5 Variance-Covariance Matrix for \((\hat{a}, \hat{b})\)

The variance-covariance matrix is useful in quantifying the variability in the estimated parameters and is obtained from Equations (2.50), (2.54), (2.55), and (2.56) by substituting \( a = \hat{a} = 1348, b = \hat{b} = 0.124 \), and the actual data values from Table 2.5. For data set DS1, we get

\[
\mathbf{E}_{\text{cov}} = \begin{pmatrix}
2368 & -0.2071 \\
-0.2071 & 5.554 \times 10^{-5}
\end{pmatrix}.
\]

From this we have
Figure 2.8. Joint confidence regions for $a$ and $b$
for Data Set DS1.
Standard Deviation \( \hat{a} \) \( \equiv \sqrt{\text{Var} \ (\hat{a})} = 48.66 \)

Standard Deviation \( \hat{b} \) \( \equiv \sqrt{\text{Var} \ (\hat{b})} = 0.00745 \)

Correlation Coefficient \( \hat{\rho}_{\hat{a},\hat{b}} \)

\[
\frac{-0.2071}{\sqrt{(2368)(5.554 \times 10^{-5})}} = -0.571 .
\]

2.8.6 Number of Remaining Errors

One useful quantity is the estimated number of remaining faults or errors in the system after some time \( t \). This value is obtained from Equation (2.19) as

\[
E(\hat{N}(t)) = \hat{\theta} \cdot e^{-bt}
\]

or

\[
E(\hat{N}(t)) = 1348e^{-0.124t} .
\]

A plot of this quantity is shown in Figure 2.9.

As expected, this value decreases with time. Also shown is a plot of the "actual" number of remaining errors

2-71
which is based on the assumption that all the errors were detected during 36 weeks of operation. It should be noted that this assumption is made for illustration purposes only and, in general, this may not be the case.

It would also be interesting to compute confidence bounds on $\overline{E}_N(t)$. Such bounds can be easily computed as follows.

Let $f(a,b)$ denote $\overline{E}_N(t)$. Then, it is well known [ROH76, ROU73] that $100(1-\alpha)\%$ confidence bounds for $f(a,b)$ are given by

$$
\{ \hat{f}(a,b) \pm t_{n-2; \alpha/2} \sqrt{V(\hat{f}(a,b))} \},
$$

(2.83)

where

$$
\hat{V}(\hat{f}(a,b)) = \left( \frac{\partial f}{\partial a} \frac{\partial f}{\partial b} \right) \Sigma \text{cov} \left( \frac{\partial f}{\partial a}, \frac{\partial f}{\partial b} \right) \bigg|_{a=a, \ b=b}
$$

(2.84)

and $t_{n-2; \alpha/2}$ is the upper $100(\alpha/2)$ percentage point of the $t$-distribution with $(n-2)$ degrees of freedom.
The 90% confidence limits for $E(\bar{N}(t))$ for data set DS1 are computed from the above equations and are plotted in Figure 2.9.

2.8.7 Software Reliability

As mentioned in Section 2.4, software reliability is a commonly used performance measure to assess how reliable the system is at various times. To compute software reliability, we use Equation (2.36) and get

$$\hat{R}_{X|S_{k-1}}(x|s) = e^{-\hat{a}(e^{-b}s - e^{-b}(s+x))}.$$ 

This gives the reliability after time $x$ starting from the current time $s$. For example, starting from $s = 15$, the reliability after 0.04 weeks, i.e., at $s+x = 15.04$, is

$$\hat{R}(0.04|s=15) = e^{-1348(e^{-(0.124)15} - e^{-(0.124)(15.04)})}$$

or

$$\hat{R}(15.04) = 0.354.$$
Figure 2.9. Expected number of remaining software errors and related quantities for various t (Data Set DS1)
To see how reliability varies with time, a plot of \( R(x|s=15) \) is shown in Figure 2.10.

To obtain confidence bounds on reliability, we use a procedure similar to the one used for getting bounds on \( E(\tilde{N}(t)) \). Let \( \hat{g}(a,b) \) represent \( R(x|s=15) \). Then the confidence bounds are given by

\[
\{ \hat{g}(a,b) \pm t_{n-2; \alpha/2} \sqrt{\hat{V}(\hat{g}(a,b))} \},
\]

(2.85)

where

\[
\hat{V}(\hat{g}(a,b)) = \left( \frac{\partial \hat{g}}{\partial a} \frac{\partial \hat{g}}{\partial b} \right) \text{cov} \left( \begin{array}{c} \frac{3\hat{g}}{3a} \\ \frac{3\hat{g}}{3b} \end{array} \right) \bigg|_{\hat{a}=a, \hat{b}=b}
\]

(2.86)

90% confidence bounds computed from these equations for the given data are shown in Figure 2.10.

Analyses similar to those for data set DS1 were undertaken for data sets DS2, DS3, and DS4 of Table 2.5. A summary of the results is given in Table 2.7.
Figure 2.10. Reliability and 90% confidence bounds after 15 weeks of testing
### TABLE 2.7
A SUMMARY OF DATA ANALYSES

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Data Set</th>
<th>DS1</th>
<th>DS2</th>
<th>DS3</th>
<th>DS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\hat{a}$</td>
<td></td>
<td>1348</td>
<td>1823</td>
<td>3958</td>
<td>3446</td>
</tr>
<tr>
<td>$\hat{b}$</td>
<td></td>
<td>0.124</td>
<td>0.112</td>
<td>0.0768</td>
<td>0.0771</td>
</tr>
<tr>
<td>$\sqrt{\text{Var}(\hat{a})}$</td>
<td></td>
<td>48.7</td>
<td>62.2</td>
<td>147.3</td>
<td>136.6</td>
</tr>
<tr>
<td>$\sqrt{\text{Var}(\hat{b})}$</td>
<td></td>
<td>0.00745</td>
<td>0.00643</td>
<td>0.00460</td>
<td>0.00492</td>
</tr>
<tr>
<td>$\rho_{a,b}$</td>
<td></td>
<td>-0.571</td>
<td>-0.648</td>
<td>-0.856</td>
<td>-0.855</td>
</tr>
<tr>
<td>Estimated Number of Remaining Errors at the end of Operational Demonstration</td>
<td></td>
<td>209</td>
<td>338</td>
<td>1212</td>
<td>1050</td>
</tr>
<tr>
<td>Number of Errors Detected During Nine Months of Operation</td>
<td></td>
<td>198</td>
<td>263</td>
<td>540</td>
<td>475</td>
</tr>
</tbody>
</table>
2.9 ANALYSIS OF FAILURE DATA FROM COMMAND AND CONTROL SYSTEMS

In this section, we analyze software failure data from two real-time command and control systems, SYS1 and SYS2. These data sets were reported in [MUS80] and represent failures observed during the system test phase. The number of delivered object instructions for SYS1 was 21,700 and for SYS2, 27,700. The number of programmers for SYS1 and SYS2 was 9 and 5, respectively.

For the first system, a total of 136 failures were observed over 25 hours of execution time and for the second system, the number of failures was 54 over 31 hours of execution time. The observed number of failures per execution hour and the cumulative failures are given in Table 2.8. The number of failures per hour are plotted in Figures 2.11 and 2.12, respectively. The parameters $a$ and $b$ were estimated using Equations (2.65) and (2.66) of Section 2.5 and are

\[
\begin{align*}
\text{SYS1} & \quad \hat{a} = 142.32 \quad \hat{b} = 0.125 \\
\text{SYS2} & \quad \hat{a} = 56.81 \quad \hat{b} = 0.097
\end{align*}
\]
TABLE 2.8

FAILURES IN ONE HOUR (EXECUTION TIME) INTERVALS AND CUMULATIVE FAILURES

<table>
<thead>
<tr>
<th>Hour</th>
<th>SYS1 No.</th>
<th>SYS1 Cum.</th>
<th>SYS2 No.</th>
<th>SYS2 Cum.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>27</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>16</td>
<td>43</td>
<td>6</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>11</td>
<td>54</td>
<td>4</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
<td>64</td>
<td>5</td>
<td>25</td>
</tr>
<tr>
<td>5</td>
<td>11</td>
<td>75</td>
<td>2</td>
<td>27</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>82</td>
<td>1</td>
<td>28</td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>84</td>
<td>1</td>
<td>29</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>89</td>
<td>1</td>
<td>30</td>
</tr>
<tr>
<td>9</td>
<td>3</td>
<td>92</td>
<td>0</td>
<td>30</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>93</td>
<td>1</td>
<td>31</td>
</tr>
<tr>
<td>11</td>
<td>4</td>
<td>97</td>
<td>3</td>
<td>34</td>
</tr>
<tr>
<td>12</td>
<td>7</td>
<td>104</td>
<td>7</td>
<td>41</td>
</tr>
<tr>
<td>13</td>
<td>2</td>
<td>106</td>
<td>1</td>
<td>42</td>
</tr>
<tr>
<td>14</td>
<td>5</td>
<td>111</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>15</td>
<td>5</td>
<td>116</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>16</td>
<td>6</td>
<td>122</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>122</td>
<td>0</td>
<td>42</td>
</tr>
<tr>
<td>18</td>
<td>5</td>
<td>127</td>
<td>4</td>
<td>46</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>128</td>
<td>0</td>
<td>46</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>129</td>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>21</td>
<td>2</td>
<td>131</td>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>22</td>
<td>1</td>
<td>132</td>
<td>0</td>
<td>48</td>
</tr>
<tr>
<td>23</td>
<td>2</td>
<td>134</td>
<td>1</td>
<td>49</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>135</td>
<td>1</td>
<td>50</td>
</tr>
<tr>
<td>25</td>
<td>1</td>
<td>136</td>
<td>1</td>
<td>51</td>
</tr>
<tr>
<td>26</td>
<td></td>
<td></td>
<td>0</td>
<td>51</td>
</tr>
<tr>
<td>27</td>
<td></td>
<td></td>
<td>1</td>
<td>52</td>
</tr>
<tr>
<td>28</td>
<td></td>
<td></td>
<td>0</td>
<td>52</td>
</tr>
<tr>
<td>29</td>
<td></td>
<td></td>
<td>1</td>
<td>53</td>
</tr>
<tr>
<td>30</td>
<td></td>
<td></td>
<td>0</td>
<td>53</td>
</tr>
<tr>
<td>31</td>
<td></td>
<td></td>
<td>1</td>
<td>54</td>
</tr>
</tbody>
</table>
FIG. 2.11 PLOT OF THE NUMBER OF FAILURES PER HOUR (SYS1)
The fitted models for the mean value function are:

SYS1
\[ \hat{m}(t) = 142.32(1 - e^{-0.125t}) \]

SYS2
\[ \hat{m}(t) = 56.81(1 - e^{-0.097t}) \]

Plots of the observed cumulative failures and expected failures (\(\hat{m}(t)\)) are shown in Figures 2.13 and 2.14 for SYS1 and SYS2, respectively.

Observed number of remaining errors and expected number of remaining errors were computed from \((\hat{a} - N(t))\) and \(\hat{a} \cdot e^{-bt}\), respectively, and are plotted in Figures 2.15 and 2.16 for SYS1 and SYS2, respectively. The 90% confidence bounds for \(\hat{m}(t)\) and \(E(\hat{N}(t))\) are also in Figures 2.13 to 2.16. From a study of these plots, it appears that the fitted models fit the data very well.

Expressions for software reliability for the two systems are obtained from Equation (2.36) as

\[ R(x|s=25) = e^{-142.32\{e^{-0.125(25)} - e^{-0.125(25+x)}\}} \]

and

\[ R(x|s=31) = e^{-56.81\{e^{-0.097(31)} - e^{-0.097(31+x)}\}}. \]
FIG. 2.13 NUMBER OF FAILURES AND 90% CONFIDENCE BOUNDS (SYS 1)
Fig. 2.14 NUMBER OF FAILURES AND 90% CONFIDENCE BOUNDS (SYS 2)
Fig. 2.15 Observed and expected no. of remaining errors with 90% confidence bounds on $E[N(x)] - SYS$. (2-85)
FIG. 2.16 OBSERVED AND EXPECTED NUMBER OF REMAINING ERROR AND 90% CONFIDENCE BOUNDS ON $E[N(x)]$ - SYS 2.
Plots of these reliability functions for SYS1 and SYS2, along with 90% confidence bounds, are given in Figures 2.17 and 2.18, respectively.
FIG. 2.17 RELIABILITY AND 90% CONFIDENCE BOUNDS - SYS 1

2-88
Fig. 2.18 Reliability and 90% confidence bounds - SYS 2.
2.10 ANALYSES OF VARIOUS TYPES OF ERRORS FROM A REAL-TIME CONTROL SYSTEM

In this section, we study the failure data from a real-time control system for a land-based radar system developed by the Raytheon Company [WIL77]. It was developed in a modular fashion (a total of 109 modules) and nearly all modules were written in JOVIAL/J3. (JOVIAL/J3 is the standard programming language for Air Force Command and Control Applications.) The rest of the modules, chiefly the Executive program, were written in Assembly language. The whole system has a total of 86,780 lines and 49,900 Assembly lines of code. The software system runs in JOVIAL, Raytheon's multiprocessor computer which consists of two identical processors (one utilized as a CPU and the other as an I/O control unit), and 81,920 words of 24-bit core memory. The software operates under the control of a highly centralized modular Executive program which supervises all real-time activity on both the CPU and IOCU. The software system features a common data base whose overall layout is defined by means of a COMPOOL. During compile time, the JOVIAL compiler creates the necessary linkages for operational programs to gain access to the data base.
Testing of the software system proceeded in three phases: unit testing of individual program modules, including the Executive program; integration (build) testing; and operational testing of the system in the field. Unit testing was carried out on a Digital System simulator rather than on the live computer in order to take advantage of the simulator's extensive debugging tools. On the other hand, integration testing, whose chief purpose was to check out control and data interfaces among program modules, was done on a real machine. Finally, operational testing was performed on a series of increasingly demanding missions designed to exercise the system and evaluate its response under various loads and physical environments. Operational missions were first rehearsed in conjunction with a mission simulator, then performed with a full hardware complement under actual field conditions.

2.10.1 Error Data

Integration testing was responsible for the largest number of Software Problem Reports (SPR's). The SPR forms were filled by anyone (systems analyst, programmer, or user of the software). SPR's were generated as soon
as an error (problem) was identified and were not de-
layed until a solution was devised and tested. The 
error data set used in validating the NHPP model was 
derived from the SPR's only during the acceptance and 
operational testing over a 22 month period during 1974-
1976. The data for the entire 38 month period will be 
analyzed in Section 3.

The error data was categorized according to the 
seriousness of the error as well as according to the 
type of error as follows.

**Seriousness of Error**

1. Critical - if the error is impeding the pro-
ject development;

2. Low - if it is not really necessary for a 
correction to be made for the current develop-
ment to proceed;

3. Improvement - if it is a suggestion for im-
provement but not necessary for satisfactory 
operation;


The number of errors for this classification is 
given in Table 2.9.
<table>
<thead>
<tr>
<th>Seriousness</th>
<th>Description</th>
<th>Actual number of errors detected during software testing (22 months)</th>
<th>̂a</th>
<th>̂b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Critical (1)</td>
<td>It is impeding project development</td>
<td>56</td>
<td>73</td>
<td>0.067</td>
</tr>
<tr>
<td>Low (2)</td>
<td>It is not really necessary for a correction to be made for the current development to proceed</td>
<td>57</td>
<td>58</td>
<td>0.209</td>
</tr>
<tr>
<td>Improvement</td>
<td>It is a suggestion for improvement but not necessary for satisfactory operation</td>
<td>139</td>
<td>142</td>
<td>0.176</td>
</tr>
<tr>
<td>Medium (4)</td>
<td>Medium severity</td>
<td>747</td>
<td>785</td>
<td>0.138</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td></td>
<td>999</td>
<td>1046</td>
<td>0.141</td>
</tr>
</tbody>
</table>
Type of Error

<table>
<thead>
<tr>
<th>Category</th>
<th>Description of Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Computational</td>
</tr>
<tr>
<td>B</td>
<td>Logic</td>
</tr>
<tr>
<td>D</td>
<td>Data Handling</td>
</tr>
<tr>
<td>L</td>
<td>User Requested Changes</td>
</tr>
<tr>
<td>M</td>
<td>Preset Data Base</td>
</tr>
<tr>
<td>P</td>
<td>Recurrent</td>
</tr>
<tr>
<td>E</td>
<td>Others, such as operating system/support software error,</td>
</tr>
<tr>
<td></td>
<td>routine/system interface errors, user interface errors,</td>
</tr>
<tr>
<td></td>
<td>unidentified errors, etc.</td>
</tr>
</tbody>
</table>

The total number of errors for these categories are given in Table 2.10.

Using the model and estimation technique of Sections 2.2 and 2.5, respectively, the estimated values of $a$ and $b$ were obtained and are also shown in Tables 2.9 and 2.10 for each category of errors. Thus, for critical errors the estimates are $\hat{a} = 73$ and $\hat{b} = 0.067$ and the fitted NHPP is

\[
P(N(t)=y) = \frac{(73(1-e^{-0.067t})^y e^{-73(1-e^{-0.067t})}}{y!},
\]

\[
y = 0, 1, \ldots
\]
<table>
<thead>
<tr>
<th>Category</th>
<th>Types of Errors</th>
<th>Actual number of errors detected during software testing (22 months)</th>
<th>( \hat{a} )</th>
<th>( \hat{b} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computational (A)</td>
<td>Errors in computing entry #, indices, and flag settings.</td>
<td>45</td>
<td>60</td>
<td>0.064</td>
</tr>
<tr>
<td></td>
<td>Incorrect/inaccurate equation used, et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logic (B)</td>
<td>Missing logic or condition test</td>
<td>178</td>
<td>186</td>
<td>0.141</td>
</tr>
<tr>
<td></td>
<td>Incorrect logic et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Handling (D)</td>
<td>Data initialization/setting error</td>
<td>165</td>
<td>169</td>
<td>0.167</td>
</tr>
<tr>
<td></td>
<td>Data location error et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>User Requested Changes (L)</td>
<td>Data related errors</td>
<td>394</td>
<td>421</td>
<td>0.125</td>
</tr>
<tr>
<td></td>
<td>Interface design poor et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Preset Data Base (M)</td>
<td>Nominal constants</td>
<td>80</td>
<td>83</td>
<td>0.152</td>
</tr>
<tr>
<td></td>
<td>Error messages</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Recurrent (P)</td>
<td>Redetected error</td>
<td>22</td>
<td>22</td>
<td>0.230</td>
</tr>
<tr>
<td></td>
<td>Duplicates of previous error et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Others (E)</td>
<td>Unidentified errors</td>
<td>115</td>
<td>117</td>
<td>0.182</td>
</tr>
<tr>
<td></td>
<td>I/O errors, et al.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOTAL</td>
<td></td>
<td>999</td>
<td>1046</td>
<td>0.141</td>
</tr>
</tbody>
</table>
Since the observed number of critical errors in 22 months is 56, this model indicates that 73 - 56 = 17 critical errors are still remaining in the system.

Plots of the actual and fitted values of the number of errors for each category are given in Figures 2.19 to 2.22, respectively. Comparing the actual and fitted curves, the NHPP model seems to provide a satisfactory description of these errors.
Figure 2.19  Actual software errors with several levels of seriousness during 22-month period of testing

2-97
Figure 2.20  The fitted mean value functions for several levels of seriousness
Figure 2.21 Actual software errors of several types during 22-month period of testing
Figure 2.22 The fitted mean value functions for several types of errors.
2.11 ANALYSIS OF FAILURE DATA FROM THE APOLLO PROJECT

Now we analyze the failure data from an on-board Apollo space flight software project developed by the Charles Stark Draper Laboratory, Inc. [RYD77] during the years 1967 to 1971. This software, with a size of 83,866 words, runs on the Apollo Guidance Computer (AGC) (designed by MIT/IL) which was used throughout all the Apollo, Skylab, Apollo-Soyuz, and F-8 Phase I programs. The purpose of the AGC was to compute guidance, targeting, navigation, and control functions for the Apollo space vehicle for all mission phases.

This software was developed by a group of guidance, navigation, and control engineers, programmers, and test engineers. The coding was done both in the assembly language of the AGC and in the interpretive language (INTERPRETER) developed for the project.

Testing and verification at the laboratory were performed using various facilities, including engineering simulation in the host computer, full scale digital simulation on the host computer, and a hybrid laboratory and system test laboratory that provided real-time execution. Several levels of testing were performed:
Level 1 tests were high order language programs run on the host computer to test algorithms. Level 2 was the AGC counterpart of these programs. Level 3 was intended to verify the operation of a complete program or routine including crew interface and realistic physical environment models. Level 4 testing was intended to verify mission phases, e.g., ascent, redezvous. Level 5 repeated the level 4 tests on the final rope which was released for manufacture. Level 6 took place after the ropes were released for manufacture and were intended to verify the program using actual mission data and the flight time-line.

The hybrid and system test laboratories were extensively used in parallel with digital simulation for level 3, 4, 5, and 6 tests. Levels 1 and 2 were performed exclusively on the digital or engineering simulators.

Changes to the software (as a result of software errors) were controlled by the following documents:

- Program Change Request (PCR)
- Program Change Notice (PCN)

2-102
The error data set which was derived from these documents was categorized according to types and is summarized in Table 2.11.

The estimates of the model parameters for each category and the total were obtained by the method of Section 2.5 and are given in Table 2.11. The likelihood surface (for total errors) is shown in Figure 2.23 and a plot of the contours of this surface in the (a-b) plane is given in Figure 2.24. From these figures, we note that the surface is really well behaved.

Plots of the observed and estimated total number of failures over the 35 month period are shown in Figures 2.25 and 2.26, respectively. Again, a comparison of the two sets of figures indicates that the model provides an excellent fit to the data.
<table>
<thead>
<tr>
<th>Category</th>
<th>Total No. of Errors</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation</td>
<td>171</td>
<td>173.95</td>
<td>0.1165</td>
</tr>
<tr>
<td>Logic</td>
<td>743</td>
<td>799.82</td>
<td>0.0765</td>
</tr>
<tr>
<td>Data Handling</td>
<td>265</td>
<td>305.25</td>
<td>0.0579</td>
</tr>
<tr>
<td>User Requested Changes</td>
<td>419</td>
<td>586.43</td>
<td>0.0358</td>
</tr>
<tr>
<td>Present Data Base</td>
<td>157</td>
<td>179.79</td>
<td>0.0590</td>
</tr>
<tr>
<td>Recurrent</td>
<td>58</td>
<td>60.52</td>
<td>0.0903</td>
</tr>
<tr>
<td>Hardware</td>
<td>871</td>
<td>1186.86</td>
<td>0.0378</td>
</tr>
<tr>
<td>Others</td>
<td>1650</td>
<td>1754.01</td>
<td>0.0807</td>
</tr>
<tr>
<td>Total</td>
<td>4337</td>
<td>4840.30</td>
<td>0.0647</td>
</tr>
</tbody>
</table>
Figure 2.24 Contours of the Likelihood Surface in the (a - b) Plane
FIGURE 2.25. OBSERVED NUMBER OF SOFTWARE ERRORS (APOLLO).
FIG. 2.26  EXPECTED NUMBER OF SOFTWARE ERRORS (APOLLO)
2.12 ANALYSIS OF DATA FROM A LARGE AVIONICS REAL-TIME SYSTEM

The software from which this error (failure) data is taken is a large avionics real-time system for DOD developed by the Boeing Aerospace Company [FRI77]. It consists of 40,640 lines of JOVIAL/J3B instructions and 84,065 assembly language instructions. This system was not developed in modular fashion.

The whole system consists of a controls and displays subsystem, a hardware test monitor, two system functions, and an executive system which schedules the former functions. The software consists of 5 major functional areas in the operational software and two functional areas in the simulation software. The software was designed so that, if one Avionic Control Unit breaks down, the system can still provide the basic functional capabilities. The simulator, which runs on two separate computers, allows testing to take place in the laboratory.

Testing of this software began with Module Verification Testing (MVT) performed by each module's developer. No Software Problem Reports (SPR's) were issued during MVT because, as far as configuration management is concerned, the software was not released yet. Upon comple-
tion of MVT, the developers released the modules for formal testing. Formal testing began with Inter-Module Compatibility Testing (IMCT) where the software was checked against its functional requirements as a total unit. Upon completion of IMCT, the software development group gave the software system to an independent system test group for System Validation Testing (SVT) where acceptance testing for quality control purposes was performed. When an error was discovered during testing, the usual procedure was to patch the program. Software errors were documented on software problem reports (SPR) while requirement errors were reported on Design Change Requests. The data set obtained for this analysis was from the two formal test phases and was both from the operational and simulation software for the first two versions (called blocks) of the software system.

Time to fix an error was calculated based on the number of days an SPR was open and an assumed 8 hour/day of equipment use to fix. This 8 hours was divided up among the errors open on any one day, and this fractional time was summed up over the days the SPR was open, to give the final total time spent fixing an error.
The error data set for the analysis was collected during the period October 1974 to August 1975 on a monthly basis. The errors were categorized into the following groups:

(1) Critical
(2) Low
(3) Improvement
(4) Medium
(5) Other

The total number of errors for each severity level over an eleven month period and the corresponding estimated values of $a$ and $b$ are given in Table 2.12. Plots of the observed and fitted number of errors are shown in Figures 2.27 and 2.28, respectively. Again, the model appears to provide a very good fit to the failure data.
### TABLE 2.12

NUMBER OF ERRORS BY SEVERITY

<table>
<thead>
<tr>
<th>Severity</th>
<th>Total Errors</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Critical</td>
<td>28</td>
<td>33.40</td>
<td>0.1657</td>
</tr>
<tr>
<td>2. Low</td>
<td>51</td>
<td>63.20</td>
<td>0.1495</td>
</tr>
<tr>
<td>3. Improvement</td>
<td>211</td>
<td>260.02</td>
<td>0.1517</td>
</tr>
<tr>
<td>4. Medium</td>
<td>357</td>
<td>501.99</td>
<td>0.1129</td>
</tr>
<tr>
<td>5. Other</td>
<td>780</td>
<td>1031.43</td>
<td>0.1283</td>
</tr>
<tr>
<td><strong>TOTAL</strong></td>
<td><strong>1427</strong></td>
<td><strong>1880.71</strong></td>
<td><strong>0.1293</strong></td>
</tr>
</tbody>
</table>
FIG. 2.27 OBSERVED NO. OF ERRORS BY SEVERITY (BOEING)
FIG. 2.28 EXPECTED NUMBER OF ERRORS BY SEVERITY.
3.1 INTRODUCTION

As discussed earlier, many stochastic models have been developed during the past ten years to describe the fault occurrence phenomenon in a large scale software system. Most of these models are based on the assumption that the time between system failures follows an exponential distribution with a parameter that depends either on the number of faults remaining in the system or on the elapsed execution or calendar time. A summary of these models and a comparative list of the features of some of these models was given in section 1.5.

All the models that have been proposed to date make an important assumption about the monotonicity of the software failure rate. In particular, it has been assumed that the software system experiences an improvement with time. In other words, the existing models assume that the software has a decreasing failure rate (DFR). However, in practice, it has been observed that many software systems first experience an increasing failure rate (during
the initial phases of integration) and then follow a decreasing failure rate.

In this section we develop a new model which incorporates this dynamic behavior of the software systems. The basic model is presented in section 3.2 and various software effectiveness measures are developed in section 3.3. Software reliability and related results are given in section 3.4. Methods for estimating the parameters of the model from software failure data are described in section 3.5. Analyses of software failure data from a large scale system and the Naval Tactical Data System are presented in sections 3.6 and 3.7, respectively.

Data sets from numerous other systems were analyzed to assess the applicability of this model. Also, goodness-of-fit tests were conducted following the method discussed in section 2.6. Details of these analyses and tests are not reported here for the sake of brevity. In all of the cases studied the model reported here was found to provide an excellent fit to the observed failure history.
3.2 MODEL DEVELOPMENT

In order to develop an appropriate model, we study the stochastic behavior of the fault detection phenomenon by focusing our attention on the number of faults detected by some arbitrary time $t$. Let $N(t)$ denote the number of faults detected by time $t$ and let $m(t)$ be the expected value of $N(t)$, i.e.,

$$m(t) = E[N(t)] .$$

(3.1)

The above function $m(t)$ is called the mean value function of the $N(t)$ process. It should be pointed out that here time $t$ can be calendar time, execution time, or any other suitable and consistent measure of time. In practice, however, we have found calendar time and CPU time as the commonly used measures.

3.2.1 Assumptions

We now consider the behavior of the software fault detection process as described by $N(t)$.

(i) There will be no faults detected at the beginning of the fault detection process, i.e., we
have $N(0) = 0$. Also, this implies

$$m(0) = 0.$$  \hfill (3.2)

(ii) It is quite obvious that the software system must contain a finite number of faults. In other words, if testing were to be continued indefinitely, the number of faults to be detected will be finite, so that the expected number of faults to be eventually found will be $m(\infty)$. Let

$$m(\infty) = a < \infty.$$  \hfill (3.3)

(iii) The faults to be detected are such that each one effects the failure occurrence phenomenon independently of others, but the rate at which each fault causes the system to fail depends on elapsed time. This can be expressed by taking the hazard rate $z(t)$ of each fault to be

$$z(t) = bct^{c-1}.$$  \hfill (3.4)
Note that the shape of this function will depend on the values of the parameters $b$ and $c$.

3.2.2 Expression for $m(t)$

Based on the above description of the fault detection process, we now develop an expression for $m(t)$. In terms of $m(t)$, the hazard rate at time $t$ is defined as

$$z(t) = \frac{m(t+\Delta t) - m(t)}{\Delta t (a - m(t))}.$$  

Substituting for $z(t)$ from Equation (3.4), we get

$$\frac{m(t+\Delta t) - m(t)}{\Delta t (a - m(t))} = bct^{c-1}.$$  (3.5)

By letting $\Delta t \to 0$ in the above equation, we get a first-order linear differential equation

$$m'(t) + bct^{c-1}m(t) = abct^{c-1}.$$  (3.6)

To solve the above equation for $m(t)$, we need to use the following results.
Lemma. If $P(t)$ and $Q(t)$ are two continuous functions of $t$, then the general solution of an equation of the form

$$y' + P(t)y = Q(t) \quad (3.7)$$

is

$$y = \frac{1}{h(t)} \int Q(t) h(t) \, dt, \quad (3.8)$$

where

$$h(t) = e^{\int P(t) \, dt}. \quad (3.9)$$

Proposition. Under the boundary condition $m(0) = 0$, the solution of equation (3.6) is given by

$$m(t) = a(1 - e^{-bt^c}). \quad (3.10)$$

Proof. Let the functions $P(t)$ and $Q(t)$ in the above Lemma be
\[ P(t) = bct^{c-1} \]

and

\[ Q(t) = abct^{c-1} . \]

Then \( h(t) \) is obtained from (3.9) as

\[ h(t) = e^{\int P(t) \, dt} \]

or

\[ h(t) = e^{bt^{c}} \quad (3.11) \]

and

\[ \int Q(t)h(t)\,dt = \int abct^{c-1}e^{bt^{c}}\,dt \]

or

\[ \int Q(t)h(t)\,dt = ae^{bt^{c}} + k , \quad (3.12) \]

where \( k \) is a constant to be determined by the boundary condition \( m(0) = 0 \). Finally, we get the solution of
(3.6) by substituting (3.11) and (3.12) into (3.8), i.e.,

\[ m(t) = e^{-bt^c}(ae^{bt^c} + k) \]

or

\[ m(t) = a + ke^{-bt^c} . \quad (3.13) \]

Since \( m(0) = 0 \), we have

\[ m(0) = a + k = 0 \]

or

\[ k = -a . \]

Substituting \( k = -a \) in (3.13), we get the result of Equation (3.10).

3.2.3 Fault Detection Rate

Fault detection rate is the number of faults per unit time. Let \( \lambda(t) \) denote the software fault detec-
tion rate so that, for a small time interval $\Delta t$, $\lambda(t)\Delta t$ represents the number of software faults detected during $(t, t+\Delta t)$. Now $m(t)$ is the expected number of faults detected by $t$ and

$$\lambda(t) = m'(t) .$$

(3.14)

From Equations (3.10) and (3.14), we get

$$\lambda(t) = abt^c - bt^c \cdot c - 1$$

or

$$\lambda(t) = at^{\gamma - 1} \cdot e^{-bt^\gamma}$$

(3.15)

where

$$\alpha = abc$$

$$\beta = b$$

$$\gamma = c$$

(3.16)

In order to see the shape of the fault detection rate $\lambda(t)$, we differentiate Equation (3.15) with respect to $t$ and equate the result to zero and get
\[ t^\gamma = \frac{\gamma - 1}{\beta^\gamma} . \quad (3.17) \]

We see that, for \( r > 1 \), \( \lambda(t) \) is a unimodal function with

\[ \left( \lambda(0) = \lambda(\omega) = 0 , \right. \]

and its maximum value occurs at \( t = t_m \) where

\[ t_m = (\frac{\gamma - 1}{\beta^\gamma})^{1/\gamma} . \quad (3.18) \]

The maximum value of \( \lambda(t) \) is

\[ \max \lambda(t) \equiv \lambda(t_m) = \alpha (\frac{\gamma - 1}{\beta^\gamma}) (\gamma - 1)/\gamma \cdot e^{-(\gamma - 1)/\gamma} . \quad (3.19) \]

In other words, the error detection rate of software or, equivalently, the software failure rate, increases during the period \( (0, t_m) \), achieves its maximum value \( \lambda(t_m) \) at \( t = t_m \), and then decreases for \( t > t_m \) eventually becoming zero at \( t = \infty \). Note that if \( 0 < \gamma < 1 \), then the software failure rate is monotonically decreasing. From the above discussion, we see that the software fault detection rate \( \lambda(t) \) is increasing/decreasing if \( r > 1 \), and monotonically decreasing if \( 0 < \gamma < 1 \).
The extreme values are $\lambda(0) = \alpha$ for $\gamma = 1$ and $\lambda(0) = \infty$ for $0 < \gamma < 1$.

3.2.4 Failure Counting Process

Now we assume that the failure counting process $N(t)$ has the following characteristics:

(i) $N(t)$ has independent increments, i.e.,

\[
\{N(t_2) - N(t_1)\} \text{ is independent of } \{N(t_3) - N(t_2)\} \text{ for some } t_1 < t_2 < t_3.
\]

(ii) The probabilities associated with the $N(t)$ process are as follows:

\[
N(t+\Delta t) - N(t) = \begin{cases} 
0 & \text{with probability } 1 - \lambda(t)\Delta t + O(\Delta t) \\
1 & \text{with probability } \lambda(t)\Delta t + O(\Delta t) \\
2 & \text{with probability } O(\Delta t)
\end{cases} \tag{3.20}
\]

It is well known that with the above properties and with $\lambda(t)$ as given in Equation (3.15), the $N(t)$ process is a non-homogeneous Poisson process (NHPP) with a mean value function $m(t)$ given in Equation (3.10). Hence, the distribution of $N(t)$ is given by

3-11
Under the assumptions discussed above, the stochastic behavior of the software failure phenomenon can be completely described by the model given in Equations (3.10) and (3.21). These equations constitute the basic failure occurrence model discussed in this section.
3.3 SOFTWARE EFFECTIVENESS MEASURES

In this section, we develop expressions for several useful quantitative measures for assessing the software system effectiveness.

3.3.1 Distribution of the Number of Faults Detected or Failures Observed

As indicated above, \( N(t) \) is a NHPP with a probability mass function

\[
P(N(t) = y) = \frac{[a(1-e^{-bt\epsilon})]^y}{y!} e^{-a(1-e^{-bt\epsilon})},
\]

\( y = 0,1,2,... \) \hspace{1cm} (3.22)

As \( t \to \infty \), we have

\[
P(N(\infty) = y) = \frac{a^y}{y!} e^{-a}, \hspace{1cm} y = 0,1,2,... \hspace{1cm} (3.23)
\]

This last expression tells us that, if the system were to be used for a long time \( (t = \infty) \), the number of faults detected or failures observed during this time follows a Poisson process with mean \( 'a' \).
3.3.2 Number of Faults Remaining in the System

Let \( \bar{N}(t) \) denote the number of faults not detected by time \( t \), i.e., the number of faults remaining in the system. Clearly, this number will be obtained by subtracting \( N(t) \) from \( N(\infty) \), the number of faults to be eventually detected. Note that these quantities are random variables. Thus, we have

\[
\bar{N}(t) = N(\infty) - N(t)
\]  

(3.24)

and

\[
E[\bar{N}(t)] = E[N(\infty)] - E[N(t)]
\]

or

\[
E[\bar{N}(t)] = \alpha - \alpha(1 - e^{-bt^C})
\]

or

\[
E[\bar{N}(t)] = \alpha e^{-bt^C}.
\]  

(3.25)
3.3.3 **Conditional Distribution of \( \bar{N}(t) \)**

If we have already observed \( y \) faults, it is useful to know the distribution of the number of faults yet to be detected. In other words, the conditional distribution of \( \bar{N}(t) \), given that \( N(t) = y \), is

\[
P(\bar{N}(t) = x | N(t) = y) = \frac{P(\bar{N}(t) = x, N(t) = y)}{P(N(t) = y)}. \tag{3.26}
\]

Now the event \( \bar{N}(t) = x \) denotes occurrences over the time interval \((t, \infty)\) while the event \( N(t) = y \) denotes occurrences over the interval \((0, t)\), i.e., these two events represent non-overlapping time intervals. From a basic property of the NHPP process, such events are independent of each other, so that we have

\[
P(\bar{N}(t) = x | N(t) = y) = P(\bar{N}(t) = x), \ x = 0, 1, 2, \ldots \tag{3.27}
\]

or

\[
P(N(\infty) - N(t) = x | N(t) = y) =
\]

\[
\frac{(m(\infty) - m(t))^x}{x!} \cdot e^{-(m(\infty) - m(t))}.
\]

Or, substituting for \( m(\infty) \) and \( m(t) \) from Equation (3.10), we get
\[ P(N(\infty) - N(t) = x | N(t) = y) = \]
\[ \frac{(a-a(1-e^{-bt^C})) x}{x!} \cdot e^{-a-a(1-e^{-bt^C})}. \]

This yields
\[ P(\bar{N}(t) = x | N(t) = y) = \frac{(ae^{-bt^C}) x}{x!} \cdot e^{-ae^{-bt^C}}. \quad (3.28) \]

Finally, the expected number of faults to be detected, given \( N(t) = y \), is
\[ E[\bar{N}(t) | N(t) = y] = ae^{-bt^C}. \quad (3.29) \]

### 3.3.4 Joint Counting Probability

The property of independent increments, along with the equations developed above, provides a complete statistical characterization of the NHPP process so that the joint probability of certain number of faults occurring in given time intervals is obtained as follows. Consider times \( t_1, t_2, \ldots, t_n \) such that \( 0 < t_1 < t_2 < \ldots < t_n \). We have, with \( t_0 = 0, y_0 = 0 \),
\[ P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n) \]

\[
= \prod_{i=1}^{n} P(N(t_i) - N(t_{i-1}) = y_i - y_{i-1}) \quad (3.30)
\]

\[
= \prod_{i=1}^{n} \frac{n \{m(t_i) - m(t_{i-1})\}^{y_i - y_{i-1}}}{(y_i - y_{i-1})!} \cdot e^{-\{m(t_i) - m(t_{i-1})\}}
\]

Equation (3.30) will be used for estimating the parameters \(a\), \(b\), and \(c\) from given failure data in later sections.
3.4 SOFTWARE RELIABILITY AND DISTRIBUTION OF TIME BETWEEN SOFTWARE FAILURES

The time between failures is a stochastic process whose behavior is governed by many factors such as the usage of the system, system load, degree of purification of software, etc. However, since it is not presently feasible to quantify the effects of these factors individually, we model the process behavior as described above, i.e., by a NHPP process with an increasing/decreasing fault detection rate. At any given point, the time to next failure will depend on the time when the last failure occurred. Suppose that the \((k-1)\)st failure occurred at some time \(S_{k-1} = s\). Then the probability that the \(k\)th failure will not occur for an additional time \(X_k = x\), i.e., the conditional probability for time \(x\), is as follows:

\[
P(\text{no failure in } (s,s+x]\mid \text{failure at } s) = R_{X_k\mid S_{k-1}}(x\mid s) = e^{-a(e^{-bs^c} - e^{-b(s+x)^c}} \quad (3.31)
\]

and

\[
R_{X_k\mid S_{k-1}}(x\mid s) = e^{-a(e^{-bs^c} - e^{-b(s+x)^c}} \quad .
\]
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS 1963-A
Since the conditional cumulative distribution function (cdf) is related to conditional reliability by

\[ F_{X_k|S_{k-1}}(x|s) = 1 - R_{X_k|S_{k-1}}(x|s), \quad (3.32) \]

we have

\[ F_{X_k|S_{k-1}}(x|s) = 1 - e^{-a(e^{-bs^c} - e^{-b(s+x)^c})}. \quad (3.33) \]

The conditional probability density function (pdf) is obtained from Equation (3.33) by differentiating \( F_{X_k|S_{k-1}}(x|s) \) with respect to \( x \) and is given by

\[ f_{X_k|S_{k-1}}(x|s) = abc(s+x)^{c-1} e^{-b(s+x)^c} e^{-a(e^{-bs^c} - e^{-b(s+x)^c})}. \quad (3.34) \]

Finally, we are also interested in the joint pdf of the cumulative times to failures, i.e., in the joint pdf of \( S_1, S_2, \ldots, S_n \). Following the approach given in Section 2, we get

\[ f_{S_1, S_2, \ldots, S_n}(s_1, s_2, \ldots, s_n) = e^{-m(s_n) n} \prod_{k=1}^{n} \lambda(s_k), \quad (3.35) \]

where
\[ m(s_n) = a(1-e^{-bs_n^c}), \quad (3.36) \]

and

\[ \lambda(s_k) = a s_k^{\gamma-1} e^{-\beta s_k^\gamma}, \quad (3.37) \]

or

\[ \lambda(s_n) = abc \cdot s_n^{c-1} \cdot e^{-bs_n^c}. \quad (3.38) \]

These results are used for estimating the parameters \( a, b, \) and \( c \) in later sections.
3.5 ESTIMATION OF MODEL PARAMETERS FROM FAILURE DATA

In this section, we describe methods for estimating the parameters \( a, b, \) and \( c \) (or, equivalently, \( \alpha, \beta, \) and \( \gamma \)) from available data on software failures. Such data are generally available either as cumulative number of failures in given time intervals, or as times between software failures. The estimation procedure is different for each case and is described below. In this report, we use the method of maximum likelihood for estimation purposes.

3.5.1 Maximum Likelihood Estimation When Data on Cumulative Software Failures are Given

Let \( y_1 \) be the number of failures observed during a time interval \((0, t_1)\), \( y_2 \) during the interval \((0, t_2)\), and so on. In general, let \( y_i \) be the number of failures by time \( t_i \). Then the observed data in this case will consist of pairs \((t_i, y_i)\), \( i = 1, 2, \ldots, n \). Now the probability of observing \( (y_1 - y_{i-1}) \) failures during a time interval \((t_i - t_{i-1})\) is given by (see Equation (3.30)),

\[
P(N(t_i) - N(t_{i-1}) = y_i - y_{i-1})
\]

\[
e^{\frac{(m(t_i) - m(t_{i-1}))}{(y_i - y_{i-1})!}(-m(t_i) - m(t_{i-1}))}
\]

\[(3.39)\]

3-21
Since the increment in the number of failures during the non-overlapping time periods \((0, t_1), (t_1, t_2), \ldots, (t_{i-1}, t_i), \ldots, (t_{n-1}, t_n)\), are independent of each other, the joint probability of the pairs of observations \((t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n)\), \(P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_i) = y_i, \ldots, N(t_n) = y_n)\), can be written as

\[
P[N(t_1) = y_1], P[N(t_2-t_1) = y_2-y_1], \ldots,
\]

\[
P[N(t_i-t_{i-1}) = y_i-y_{i-1}], \ldots,
\]

\[
P[N(t_n-t_{n-1}) = y_n-y_{n-1}]
\]

\[
= \prod_{i=1}^{n} P[N(t_i-t_{i-1}) = y_i-y_{i-1}]
\]

\[
= \prod_{i=1}^{n} \frac{n \{m(t_i)-m(t_{i-1})\}^{y_i-y_{i-1}} - \sum_{i=1}^{n} \{m(t_i)-m(t_{i-1})\}}{(y_i-y_{i-1})!} e^{i=1}
\]

or

\[
P(N(t_1) = y_1, N(t_2) = y_2, \ldots, N(t_n) = y_n)
\]

\[
= \prod_{i=1}^{n} \frac{n \{m(t_i)-m(t_{i-1})\}^{y_i-y_{i-1}} - m(t_n)}{(y_i-y_{i-1})!} e^{-m(t_n)}.
\]

(3.40)
From this, the likelihood function for parameters $a$, $b$, and $c$, corresponding to the observations $(t_i, y_i)$, $i = 1, 2, \ldots, n$, is obtained as

$$L(a, b, c | (t_1, y_1), (t_2, y_2), \ldots, (t_n, y_n))$$

$$= \prod_{i=1}^{n} \frac{(y_i - y_{i-1})!}{(y_i - y_{i-1})!} e^{-m(t_n)} \cdot (3.41)$$

Taking the natural logarithm on both sides of Equation (3.41), the log likelihood is obtained as

$$\ell(a, b, c | (t_i, y_i), i=1, 2, \ldots, n) = \ln L(a, b, c | (t_i, y_i)),$$

$$i = 1, 2, \ldots, n$$

$$= \sum_{i=1}^{n} (y_i - y_{i-1}) \ln(m(t_i) - m(t_{i-1}))$$

$$- m(t_n) - \sum_{i=1}^{n} \ln(y_i - y_{i-1})! \quad (3.42)$$

On substituting for $m(t_{i-1})$, $m(t_i)$, and $m(t_n)$ from Equation (3.10), and simplifying, the log-likelihood function becomes
\[ l(a,b,c|(t_i,y_i), i=1,2,...,n) \]
\[ = \sum_{i=1}^{n} (y_i-y_{i-1}) \ln(a(e^{-bt_{i-1}} - e^{-bt_i})) \]
\[ - bt_c^c \sum_{i=1}^{n} \ln(y_i-y_{i-1})! \]  \[ (3.43) \]

It is well known that the maximum likelihood estimates (mle's) \( \hat{a}, \hat{b}, \) and \( \hat{c}, \) are those values of \( a, b, \) and \( c, \) respectively, that maximize the likelihood function given in equation (3.42), or equivalently, are those values that maximize the log likelihood function of Equation (3.43). Thus, \( \hat{a}, \hat{b}, \) and \( \hat{c}, \) are those values that simultaneously satisfy the following equations:

\[ \frac{\partial l}{\partial a} = 0 . \]  \[ (3.44) \]
\[ \frac{\partial l}{\partial b} = 0 . \]  \[ (3.45) \]
\[ \frac{\partial l}{\partial c} = 0 . \]  \[ (3.46) \]

On taking the derivatives of Equation (3.43) with respect to \( a, b, \) and \( c, \) and substituting in Equations (3.44), (3.45), and (3.46), respectively, we get the
The set of simultaneous equations (3.47), (3.48), and (3.49) can be solved numerically for \(a\), \(b\), and \(c\). The solution will be the required maximum likelihood estimates \(\hat{a}\), \(\hat{b}\), and \(\hat{c}\) of \(a\), \(b\), and \(c\), respectively.
3.5.1.1 Variance-Covariance of $\hat{a}$, $\hat{b}$, and $\hat{c}$

Once the estimates of $a$, $b$, and $c$ have been obtained from the data, the performance measures, as derived in sections 3.3 and 3.4, can be easily computed by substituting $\hat{a}$, $\hat{b}$, and $\hat{c}$ for $a$, $b$, and $c$, respectively. In order to obtain confidence bounds on the performance measures, we need to know the distribution of the estimates $\hat{a}$, $\hat{b}$, and $\hat{c}$. For a reasonably large sample size $n$, say $n > 20$, the maximum likelihood estimators generally follow a normal distribution. Thus the vector $(\hat{a} \hat{b} \hat{c})'$ will have a trivariate normal distribution (TVN) with $(a \ b \ c)'$ as the vector of means and $\Sigma_{\text{cov}}$ as the variance-covariance matrix. In other words, for large $n$,

$$
\begin{pmatrix}
\hat{a} \\
\hat{b} \\
\hat{c}
\end{pmatrix} \sim \text{TVN} \left( \begin{pmatrix} a \\ b \\ c \end{pmatrix}, \Sigma_{\text{cov}} \right). \tag{3.50}
$$

The variance-covariance matrix $\Sigma_{\text{cov}}$ represents

$$
\Sigma_{\text{cov}} = 
\begin{pmatrix}
\text{Var}(a) & \text{Cov}(a,b) & \text{Cov}(a,c) \\
\text{Cov}(b,a) & V(b) & \text{Cov}(b,c) \\
\text{Cov}(c,a) & \text{Cov}(c,b) & V(c)
\end{pmatrix} \tag{3.51}
$$

3-26
and is given by

\[ E_{\text{cov}} = \begin{pmatrix} r_{aa} & r_{ab} & r_{ac} \\ r_{ba} & r_{bb} & r_{bc} \\ r_{ca} & r_{cb} & r_{cc} \end{pmatrix}^{-1} \]  

(3.52)

where

\[ r_{ij} = -E[\frac{\partial^2 \ell}{\partial \theta_i \partial \theta_j}], \quad i, j = a, b, c. \]  

(3.53)

Thus, to obtain \( E_{\text{cov}} \), we first take the derivative of Equation (3.43) and then the expectations of the resulting expressions, as indicated by Equation (3.53). On so doing, we get the following expressions for various \( r_{ij} \)'s, \( i, j = a, b, c \).

\[ r_{aa} = \frac{1}{a} \sum_{i=1}^{n} (e^{-bt_i} - 1 - e^{-bt_i}) \]  

(3.54)

\[ r_{ab} = r_{ba} = t_n^c e^{-bt_n} \]  

(3.55)

\[ r_{ac} = r_{ca} = bt_n^c (tnt_n) e^{-bt_n} \]  

(3.56)
\[ r_{bb} = a \sum_{i=1}^{n} \frac{(t_{i-1}^c + t_i^c)^2 e^{-bt_{i-1}^c} - b(t_{i-1}^c + t_i^c)}{e^{-bt_i^c} - e^{-bt_{i-1}^c}} + at_{n}^2 e^{-bt_{n}^c} \] (3.57)

\[ r_{bc} = r_{cb} = a \sum_{i=1}^{n} \frac{\left( \frac{t_{i}^c \cdot t_{i-1}^c}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \right) \cdot \left( \frac{t_{i}^c \cdot t_{i-1}^c}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \right) \cdot \left( \frac{t_{i}^c \cdot t_{i-1}^c}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \right) \cdot \left( \frac{t_{i}^c \cdot t_{i-1}^c}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \right) \cdot \left( \frac{t_{i}^c \cdot t_{i-1}^c}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \right)}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \] (3.58)

\[ r_{cc} = ab \sum_{i=1}^{n} \frac{b(l-bt_{i}^c)^2 \left[ t_{i}^c \cdot (e^{-bt_{i}^c} - e^{-bt_{i-1}^c}) \right]^2}{e^{-bt_{i}^c} - e^{-bt_{i-1}^c}} \] (3.59)
The variance-covariance matrix $\Sigma_{\text{cov}}$ is obtained by substituting the appropriate values from Equations (3.54) to (3.59) into Equation (3.53). Confidence bounds on the performance measures can then be computed by using the properties of a trivariate normal distribution.

3.5.2 Maximum Likelihood Estimation of Parameters When Data on Times Between Software Failures are Given

Sometimes failure data are given as a sequence of failure times $s_1, s_2, \ldots, s_n$ where $s_k$, $k = 1, 2, \ldots, n$, represents the time of the $k$th failure. Using the joint density of $S_1, S_2, \ldots, S_n$, as given in Equation (3.35), the likelihood function of $a$, $b$, and $c$ for given data $s_1, s_2, \ldots, s_n$ is

$$L(a, b, c | s_1, s_2, \ldots, s_n) = e^{-a(1-e^{-b s_n}) n} \prod_{k=1}^{abc}{s_k^{c-1}e^{-b s_k}}.$$  \hspace{1cm} (3.60)

As before, the maximum likelihood estimates are those values which maximize the likelihood function of Equation (3.60). Since maximizing the likelihood is equivalent to maximizing the log-likelihood function, we take the natural logarithm of Equation (3.60) and get

3-29
\[ \ell(a, b, c | s_1, s_2, \ldots, s_n) = \ln L(a, b, c | s_1, s_2, s_n) \]
\[ = n \ln a + n \ln b + n \ln c + (c-1) \sum_{k=1}^{n} \ln s_k \]
\[ - b \sum_{k=1}^{n} s_k - a(1 - e^{-n}) . \]  \hspace{1cm} (3.61)

Then, the mle's are those values \( \hat{a}, \hat{b}, \hat{c} \) which satisfy the following equations:

\[ \frac{\partial \ell}{\partial a} = 0 , \]  \hspace{1cm} (3.62)

\[ \frac{\partial \ell}{\partial b} = 0 , \]  \hspace{1cm} (3.63)

\[ \frac{\partial \ell}{\partial c} = 0 . \]  \hspace{1cm} (3.64)

On taking the derivatives of Equations (3.61), (3.62), (3.63), and (3.64), respectively,

\[ n = a(1 - e^{-n}) , \]  \hspace{1cm} (3.65)

\[ n = b\left( \sum_{k=1}^{n} s_k^c + a s_c^c e^{-n} \right) , \]  \hspace{1cm} (3.66)

and

3-30
\[ n + c \sum_{k=1}^{n} \ln s_k = b \left( \sum_{k=1}^{n} s_k^c \ln s_k + a^C S_n \right) e^{-bs^C_n}. \quad (3.67) \]

The above simultaneous, non-linear equations can be solved numerically to get the maximum likelihood estimates \( \hat{a} \), \( \hat{b} \), and \( \hat{c} \).

Since the joint distribution of \( (S_1, S_2, \ldots, S_n) \) is an improper distribution, as discussed in section 2, the asymptotic properties of the mle's do not hold in this case.
3.6 ANALYSIS OF FAILURE DATA FROM A LARGE SCALE SOFTWARE SYSTEM

Failure data generated during formal testing of a large scale software system [THA76] was analyzed in section 2.8 using a two parameter non-homogeneous Poisson process model. In that analysis, the data from the first 9 of the 24 weeks of testing had to be dropped because during this period the system exhibited an increasing failure rate. The model developed in this section is capable of modelling an increasing/decreasing failure rate and will be employed to develop a model for the failure data over the entire 24 week testing period.

The number of failures per week for the four data sets are given in Table 3.1 and a plot for data set DS1 is shown in Figure 3.1. It is readily seen that the failure rate increases for about the first nine weeks and then decreases until the end of testing.

3.6.1 Estimation of Parameters

The data are given in the form of point \((t_i, y_i)\), \(i = 1, 2, \ldots, 24\), where \(t_i\) and \(y_i\) refer to time in weeks and \(y_i\) is the number of failures in week \(i\). To esti-
TABLE 3.1. Software Failure Data From Thayer et al. [THAT76].

<table>
<thead>
<tr>
<th>WEEK</th>
<th>DS1 # OF SPR's</th>
<th>CUMULATIVE</th>
<th>DS2 # OF SPR's</th>
<th>CUMULATIVE</th>
<th>DS3 # OF SPR's</th>
<th>CUMULATIVE</th>
<th>DS4 # OF SPR's</th>
<th>CUMULATIVE</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>67</td>
<td>67</td>
<td>69</td>
<td>69</td>
<td>79</td>
<td>79</td>
<td>77</td>
<td>77</td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>173</td>
<td>111</td>
<td>180</td>
<td>144</td>
<td>223</td>
<td>139</td>
<td>216</td>
</tr>
<tr>
<td>3</td>
<td>97</td>
<td>270</td>
<td>99</td>
<td>279</td>
<td>141</td>
<td>364</td>
<td>139</td>
<td>355</td>
</tr>
<tr>
<td>4</td>
<td>129</td>
<td>399</td>
<td>133</td>
<td>412</td>
<td>199</td>
<td>563</td>
<td>195</td>
<td>550</td>
</tr>
<tr>
<td>5</td>
<td>77</td>
<td>476</td>
<td>87</td>
<td>499</td>
<td>143</td>
<td>706</td>
<td>133</td>
<td>683</td>
</tr>
<tr>
<td>6</td>
<td>143</td>
<td>619</td>
<td>156</td>
<td>655</td>
<td>233</td>
<td>939</td>
<td>220</td>
<td>903</td>
</tr>
<tr>
<td>7</td>
<td>116</td>
<td>735</td>
<td>131</td>
<td>786</td>
<td>202</td>
<td>1141</td>
<td>187</td>
<td>1090</td>
</tr>
<tr>
<td>8</td>
<td>131</td>
<td>866</td>
<td>141</td>
<td>927</td>
<td>203</td>
<td>1344</td>
<td>193</td>
<td>1283</td>
</tr>
<tr>
<td>9</td>
<td>187</td>
<td>1053</td>
<td>211</td>
<td>1158</td>
<td>316</td>
<td>1660</td>
<td>292</td>
<td>1575</td>
</tr>
<tr>
<td>10</td>
<td>203</td>
<td>1256</td>
<td>238</td>
<td>1376</td>
<td>369</td>
<td>2029</td>
<td>334</td>
<td>1909</td>
</tr>
<tr>
<td>11</td>
<td>136</td>
<td>1392</td>
<td>179</td>
<td>1555</td>
<td>298</td>
<td>2327</td>
<td>255</td>
<td>2164</td>
</tr>
<tr>
<td>12</td>
<td>183</td>
<td>1575</td>
<td>222</td>
<td>1777</td>
<td>398</td>
<td>2725</td>
<td>359</td>
<td>2523</td>
</tr>
<tr>
<td>13</td>
<td>47</td>
<td>1622</td>
<td>73</td>
<td>1850</td>
<td>115</td>
<td>2840</td>
<td>89</td>
<td>2612</td>
</tr>
<tr>
<td>14</td>
<td>46</td>
<td>1668</td>
<td>56</td>
<td>1906</td>
<td>83</td>
<td>2923</td>
<td>73</td>
<td>2685</td>
</tr>
<tr>
<td>15</td>
<td>71</td>
<td>1739</td>
<td>88</td>
<td>1994</td>
<td>150</td>
<td>3073</td>
<td>133</td>
<td>2818</td>
</tr>
<tr>
<td>16</td>
<td>54</td>
<td>1793</td>
<td>78</td>
<td>2072</td>
<td>142</td>
<td>3215</td>
<td>118</td>
<td>2936</td>
</tr>
<tr>
<td>17</td>
<td>57</td>
<td>1850</td>
<td>92</td>
<td>2164</td>
<td>172</td>
<td>3387</td>
<td>137</td>
<td>3073</td>
</tr>
<tr>
<td>18</td>
<td>80</td>
<td>1930</td>
<td>104</td>
<td>2268</td>
<td>202</td>
<td>3589</td>
<td>178</td>
<td>3251</td>
</tr>
<tr>
<td>19</td>
<td>64</td>
<td>1994</td>
<td>85</td>
<td>2353</td>
<td>168</td>
<td>3757</td>
<td>147</td>
<td>3398</td>
</tr>
<tr>
<td>20</td>
<td>27</td>
<td>2021</td>
<td>53</td>
<td>2406</td>
<td>89</td>
<td>3846</td>
<td>63</td>
<td>3461</td>
</tr>
<tr>
<td>21</td>
<td>42</td>
<td>2063</td>
<td>45</td>
<td>2451</td>
<td>87</td>
<td>3933</td>
<td>84</td>
<td>3545</td>
</tr>
<tr>
<td>22</td>
<td>55</td>
<td>2118</td>
<td>59</td>
<td>2510</td>
<td>111</td>
<td>4044</td>
<td>107</td>
<td>3652</td>
</tr>
<tr>
<td>23</td>
<td>62</td>
<td>2180</td>
<td>84</td>
<td>2594</td>
<td>253</td>
<td>4297</td>
<td>231</td>
<td>3883</td>
</tr>
<tr>
<td>24</td>
<td>11</td>
<td>2191</td>
<td>27</td>
<td>2621</td>
<td>70</td>
<td>4367</td>
<td>54</td>
<td>3937</td>
</tr>
</tbody>
</table>
Fig. 3.1 A plot of the observed number of failures per week for Data Set DS1.
mate the parameters a, b, and c, we use the method of section 3.5.1 and substitute the data values for each set in Equations (3.47), (3.48), and (3.49). The estimates \( \hat{a} \), \( \hat{b} \), and \( \hat{c} \) for the four sets are given in Table 3.2, and the fitted mean value functions for the four data sets are as follows

\[
\begin{align*}
DS1: \quad \hat{m}(t) &= 2352(1 - e^{-0.0232t^{1.494}}) \\
DS2: \quad \hat{m}(t) &= 2873(1 - e^{-0.0182t^{1.540}}) \\
DS3: \quad \hat{m}(t) &= 5182(1 - e^{-0.0135t^{1.547}}) \\
DS4: \quad \hat{m}(t) &= 4657(1 - e^{-0.0156t^{1.505}})
\end{align*}
\]

A plot of the cumulative number of observed software failures is given in Figure 3.2 and the expected cumulative number of failures (\( \hat{m}(t) \)) for each data set are shown in Figure 3.3.
### Table 3.2
A Summary of Data Analysis for DS1-DS4

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Data Set</th>
<th>DS1</th>
<th>DS2</th>
<th>DS3</th>
<th>DS4</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \hat{a} )</td>
<td></td>
<td>2352</td>
<td>2873</td>
<td>5182</td>
<td>4657</td>
</tr>
<tr>
<td>( \hat{b} )</td>
<td></td>
<td>0.0232</td>
<td>0.0182</td>
<td>0.0135</td>
<td>0.0156</td>
</tr>
<tr>
<td>( \hat{c} )</td>
<td></td>
<td>1.494</td>
<td>1.540</td>
<td>1.547</td>
<td>1.505</td>
</tr>
<tr>
<td>( \sqrt{\text{Var}(\hat{a})} )</td>
<td></td>
<td>55.4</td>
<td>65.3</td>
<td>118.0</td>
<td>110.5</td>
</tr>
<tr>
<td>( \sqrt{\text{Var}(\hat{b})} )</td>
<td></td>
<td>0.00188</td>
<td>0.00143</td>
<td>0.00086</td>
<td>0.00101</td>
</tr>
<tr>
<td>( \sqrt{\text{Var}(\hat{c})} )</td>
<td></td>
<td>0.0354</td>
<td>0.0345</td>
<td>0.0297</td>
<td>0.0304</td>
</tr>
<tr>
<td>( \hat{\rho}_{a,b} )</td>
<td></td>
<td>0.184</td>
<td>0.224</td>
<td>0.286</td>
<td>0.273</td>
</tr>
<tr>
<td>( \hat{\rho}_{a,c} )</td>
<td></td>
<td>-0.320</td>
<td>-0.391</td>
<td>-0.584</td>
<td>-0.579</td>
</tr>
<tr>
<td>( \hat{\rho}_{b,c} )</td>
<td></td>
<td>-0.916</td>
<td>-0.914</td>
<td>-0.876</td>
<td>-0.868</td>
</tr>
<tr>
<td>Number of Errors Detected (Observed) During Operational Demonstration Period</td>
<td></td>
<td>198</td>
<td>263</td>
<td>540</td>
<td>475</td>
</tr>
<tr>
<td>Estimate of the Number of Errors to be Detected During Operational Demonstration Period</td>
<td></td>
<td>161</td>
<td>252</td>
<td>812</td>
<td>717</td>
</tr>
</tbody>
</table>

3-36
FIGURE 3.2. Plots of the Cumulative Number of Software Failures for DS1 to DS4.
FIGURE 3.3. Plots of the Expected Cumulative Number of Software Failures ($m(t)$) for DS1 to DS4.
3.6.2 Confidence Bounds

By using normal approximation to a Poisson in Equation (3.22) we compute the 90% confidence bounds for the $N(t)$ process. The estimated mean value function and 90% bounds for data set DS1 for the $N(t)$ process are plotted in Figure 3.4. From this figure we see that most of the observed points fall within 90% bounds implying that the model described in Section 3.2 fits the entire history of software errors very well. The number of remaining errors at $t = 24$ (weeks), given that 2191 errors were found by this time, is estimated from Equation (3.25) and we have

$$\mathbb{E}[\hat{N}(24)|N(24) = 2191] = 2352e^{-0.0232(24)^{1.494}} = 161.9.$$ 

Note that a total of 198 errors were detected during the one year period of operational demonstration so that the predicted number is close to the actual value. The variance-covariance matrix is obtained from Equation (3.51) and is

$$\hat{\Sigma}_{\text{cov}} = \begin{bmatrix}
3067 & 0.0191 & -0.627 \\
0.0191 & 3.53 \times 10^{-6} & -6.09 \times 10^{-5} \\
-0.627 & -6.09 \times 10^{-5} & 1.25 \times 10^{-3}
\end{bmatrix}$$
FIGURE 3.4. Estimated Mean Value Function and 90% Confidence Bounds for the N(t) Process (DS1).
From this matrix we obtain the estimated standard deviations and the appropriate correlation coefficients for \( \hat{a} \), \( \hat{b} \), and \( \hat{c} \). These values for data sets DS1 to DS4 are also shown in Table 3.2. By using the above variance-covariance matrix we can also obtain 100(1-\( \alpha \))% confidence bounds for \( \bar{E}_N(t) \) which are given by

\[
\{ \hat{f}(a, b, c) \pm t_{n-3; \alpha/2} \sqrt{\text{Var}(\hat{f}(a, b, c))} \}
\]

where

\[
\text{Var}(\hat{f}(a, b, c)) = \left( \frac{\partial f}{\partial a} \frac{\partial f}{\partial b} \frac{\partial f}{\partial c} \right) \text{cov} \left( \begin{array}{ccc}
\frac{\partial f}{\partial a} \\
\frac{\partial f}{\partial b} \\
\frac{\partial f}{\partial c}
\end{array} \right) 
\]

For this case we have

\[
\frac{\partial f}{\partial a} = e^{-bt^c}
\]

\[
\frac{\partial f}{\partial b} = -at^ce^{-bt^c}
\]

\[
\frac{\partial f}{\partial c} = -abt^c(\ln t)e^{-bt^c}
\]
The 90% confidence bounds for $\hat{E}(t)$ for data set DS1 are computed from the above equations and are shown in Figure 3.5. Also shown is a plot of the actual number of remaining errors during the 24 week period. From this figure we see that the actual errors fall within the 90% bounds.

Similarly, by setting

$$\hat{f}(a,b,c) \equiv R_{X_k|S_{k-1}}(x|s)$$

$$= e^{-a\{e^{-bs^c} - e^{-b(s+x)^c}\}}$$

we can estimate the software reliability for given debugging time $s$.

The 100(1-$a$)% confidence bounds on $R_{X_k|S_{k-1}}(x|s)$ can be obtained as for $\hat{E}(t)$. The reliability plots and 90% confidence bounds for DS1 are shown in Figure 3.6.
$a = 2352$
$b = 0.0232$
$c = 1.494$

FIGURE 3.6. Reliability Function and 90% Confidence Bounds (DSL).
3.7 ANALYSIS OF FAILURE DATA FROM NAVAL TACTICAL DATA SYSTEM (NTDS)

Failure data from NTDS were analyzed in Section 2.7 using a two parameter NHPP model. In this section we reanalyze the same data by using the three parameter NHPP model of Section 3.2. (For details of the system and data set, see Section 2.7.) Data analysis using the Newton-Raphson method for solving the likelihood estimates of $a$, $b$, and $c$ based on the first 26 failures of Table 2.1, we get $\hat{a} = 27.2$, $\hat{b} = 0.000783$, and $\hat{c} = 1.50$ so that

$$m(t) = \hat{a}(1-e^{-\hat{b}t\hat{c}})$$

$$= 27.2(1-e^{-0.000783t^{1.5}}).$$

The bounds of the $N(t)$ process can be obtained by using normal approximation to a Poisson distribution of Equation (3.22). The estimated mean value function and 90% bounds of the $N(t)$ process for this data set are shown in Figure 3.7. Also shown is a plot of the actual number of errors detected by time $t$. From this figure we see that all the data points fall within the 90% bounds. We can estimate the expected number of errors remaining
NTDS (N=26)

\[ a = 27.2 \]
\[ b = 0.000783 \]
\[ c = 1.50 \]

FIGURE 3.7. Estimated Mean Value Function and 90% Confidence Bounds for the N(t) Process (Data Set NTDS).
at time \( t \) by substituting the mle's in Equation (3.25), i.e.,

\[
\hat{E}(t) = ae^{-bt^c}
\]

or

\[
\hat{E}(t) = 27.2e^{-0.000783t^{1.5}}
\]

Thus for \( t = 250 \),

\[
\hat{E}(250) = 1.23.
\]

That is, we can expect one more error remaining at \( t = 250 \) (days). The conditional reliability of the time to the next (27th) failure, given \( S_{26} = 250 \), is computed as

\[
\hat{R}_{x|27}(x|250) = e^{-a\{e^{-b(250^c)} - e^{-b(250+x)^c}\}}
\]

\[
= e^{-27.2\{0.0453 - e^{-0.000783(250+x)^{1.5}}\}}.
\]

For the values of \( x = 10, 20, \) and \( 50 \) (days) the reliability values are 0.81, 0.68, and 0.46, respectively.
SECTION 4

OPTIMUM SOFTWARE RELEASE TIME

4.1 INTRODUCTION

An important objective of developing the models in Sections 2 and 3 was to provide an analytical framework for estimating software performance measures which are needed for making various decisions. An important decision of practical concern is the determination of the time when testing can stop and the system be considered ready for release, that is, the determination of the software release time.

The operational performance of a software system is to a large extent dependent on the time spent in testing. The longer the testing phase, the better the performance. Also, the cost of fixing an error is generally much less during testing than during operation. However, the time spent in testing delays the release of the system for operational use and incurs additional cost. This suggests a reduction in test time and an early release of the system. In this section, we consider these conflicting objectives in the determination of the optimum release time.
In Section 4.2 we consider the release time problem based on a reliability criterion using the model of Section 2. Cost based optimum release time policies are developed in Sections 4.3 and 4.4 when the failure phenomenon follows a non-homogeneous Poisson process. The policy in Section 4.3 uses the model of Section 2 while the policy in Section 4.4 is for the failure model of Section 3.
4.2 SOFTWARE RELEASE TIME BASED ON RELIABILITY CRITERION

For a non-homogeneous Poisson process failure model, the conditional reliability at operational time \(x\), given that the testing has proceeded for \(S = s\) time units, is given by

\[
R_x|S(x|s) = R = \exp[-a(e^{-bs} - e^{-b(s+x)})] \tag{4.1}
\]

or

\[
R = \exp[-m(x)e^{-bs}], \tag{4.2}
\]

where

\[
m(x) = a(1 - e^{-bx}) . \tag{4.3}
\]

One commonly used criterion is to stop testing when the predicted reliability at a specified time \(x\) equals some given value. Then the problem reduces to solving (4.2) to find the value of \(s\) that satisfies this criterion.

Taking the logarithm of both sides of (4.2) and rearranging yields
\[ s = (1/b)[\ln m(x) - \ln \ln 1/R] \]  \hspace{1cm} (4.4)

for the software system under test. In (4.3) and (4.4), \( a \) and \( b \) are estimated from previous data and \( R \) and \( x \) are the specified values. Therefore, the required testing time \( s \) can be easily determined.

For illustration purposes, consider the failure data DS1 discussed in Section 2.8. For this data set \( a = 1348 \) and \( b = 0.124 \). Suppose it is desired that the testing be continued until the operational reliability at \( x = 0.1 \) equals 0.70. From (4.4),

\[ s = \frac{1}{0.124}(\ln[1348(1-e^{-0.124(0.1)})]) \]

\[ - \ln \ln(1/0.7), \]

or \( s = 31 \) weeks.

In other words, 31 weeks of testing will be needed before the system can be released to assure the desired reliability.

To see the effect of \( s \) on \( R(x|s) \), plots of reliability versus \( bs \) for \( m(x) = 5(5)50 \) are shown in Figure 4.1. We note that, as the testing time \( s \) is increased, while keeping \( x \), and hence \( m(x) \), constant, \( R(x|s) \) in-
Fig. 4.1 Plots of reliability versus bs for $m(x) = 5(5)50$. 

4-5
creases very rapidly to approximately 0.95. After that, the increase in reliability is very slow, which indicates that a long testing time is required to get a highly reliable software system.
4.3 OPTIMUM RELEASE TIME BASED ON COST CRITERION
(MODEL OF SECTION 2)

To determine the optimal policy, we first develop a cost model and then solve it to get the desired result.

4.3.1 Cost Model and Optimal Policy

Let

\[ c_1 = \text{cost of fixing an error during testing}, \]
\[ c_2 = \text{cost of fixing an error during operation} \]
\[ (c_2 > c_1), \]
\[ c_3 = \text{cost of testing per unit time}, \]
\[ t = \text{software life-cycle length}, \text{and} \]
\[ T = \text{software release time (same as testing time)}. \]

Since \( m(t) \) represents the expected number of errors during \((0,t)\), the expected costs of fixing errors during the testing and the operational phases are \( c_1 m(T) \) and \( c_2 [m(t) - m(T)] \), respectively. Further, the testing cost during \( T \) is \( c_3 T \).
Combining the above costs, the total expected cost is given by

\[ C(T,t) = C(T) = c_1 m(T) + c_2 [m(t) - m(T)] + c_3 T. \]  \hfill (4.5)

These costs are also shown in Figure 4.2.

Our objective is to find the optimum value \( T^* \) that minimizes (4.5). Differentiating (4.5) with respect to \( T \), we get

\[ \frac{dC(T)}{dT} = c_1 m'(T) - c_2 m'(T) + c_3. \]  \hfill (4.6)

Equating the right-hand side of (4.6) with zero and noting that \( \lambda(T) \equiv m'(T) \), we get

\[ \lambda(T) = \frac{c_3}{c_2 - c_1}, \]  \hfill (4.7)

where

\[ \lambda(T) = ab e^{-bT}. \]  \hfill (4.8)

Note that \( \lambda(T) \) is a monotonically decreasing function of \( T \) and \( \lambda(0) = ab \). If \( ab \leq \frac{c_2}{c_2 - c_1} \), (4.7) has no feasible solution and, for \( T \geq 0 \), \( \frac{dC(T)}{dT} > 0 \) (see
Cost of fixing an error $C_2$

$C_2 = m(T) - m(T) + C_3 T$

Software life cycle

Fig. 4.2. Cost Components
Figure 4.3. Hence, for this case, the minimum of $C(T)$ is at $T = 0$; that is, $T^* = 0$.

Now, if $ab > c_3/(c_2 - c_1)$, there exists a unique feasible solution of (4.7) given by (see Figure 4.3)

$$T_0 = \frac{ab(c_2 - c_1)}{(1/b) \cdot n}$$

Since $dC(T)/dT < 0$ for $0 < T < T_0$ and $dC(T)/dT > 0$ for $T > T_0$, the minimum of $C(T)$ is at $T = T_0$ for $T_0 \leq T$ and at $T = t$ for $T_0 > t$. These can be summarized as follows.

**Theorem 4.1.** (i) If $ab > c_3/(c_2 - c_1)$, then there exists a unique feasible solution of (4.7) and the optimum release time is

$$T^* = \min\{T_0, t\},$$

where $T_0$ is given by (4.9).

(ii) If $ab \leq c_3/(c_2 - c_1)$, then $T^* = 0$.

It should be noted that, if the minimum expected cost exceeds the operational benefit to be gained, no testing should be undertaken.
Fig. 4.3. Plots of $\lambda(T)$ versus $T$. 

$\lambda(T)$ vs $T$ with $a b$, $\frac{c_3}{c_2-c_1}$, $T^*$, $T_0$, and $t$ marked.
To illustrate the above results, consider the data set mentioned in Section 4.2. Here \(a = 1348\) and \(b = 0.124\). Let \(c_1 = 1\), \(c_2 = 5\), \(c_3 = 100\), and \(t = 100\).

Then \(ab = 1348(0.124) = 167\) and

\[
\frac{c_3}{(c_2 - c_1)} = 25.
\]

Since \(ab > \frac{c_3}{(c_2 - c_1)}\), the optimum release time

\[
T^* = \min\{(1/0.124)\ln(167/25), 100\}
\]

or

\[
T^* = \min\{15.3, 100\} = 15.3.
\]

Hence, the optimum solution for this case is to allocate 15.3 weeks for testing and 84.7 weeks for operation. The cost associated with this policy will be \(C(T^*) = 3687\).

4.3.2 Sensitivity Analysis of \(T^*\)

Now we investigate the effects of the parameters \(a\), \(b\), and \(c_3/(c_2 - c_1)\) on the optimum release time.
First, from Theorem 4.1, we note that $T^*$ equals 0, $t$, or $T_0$. Since $T^* = 0$ and $T^* = t$ are degenerate cases, we shall consider only the case when $T^* = T_0$.

From (4.9), we see that $T_0$ increases logarithmically with $a$ and decreases logarithmically with $c_r$ as others are kept constant. Next, the first and second derivatives of $T_0$ with respect to $b$ indicate that $T$ is a concave function of $b$ with maximum at $b_0 = ec_r/a$, and the maximum value of $T_0$ is $1/b_0$.

In practice, for a given software system, the value of $a$ prior to testing is fixed and one may be interested in the joint effect of $b$ and $c_r$ on $T_0$. The value of $b$ can be affected by an appropriate selection of testing strategies and techniques. For the data set discussed earlier, $a = 1348$. For this case, contours of $T_0$ in the $b$-$c_r$ plane are shown in Figure 4.4. Also shown is the optimum value of $T$ corresponding to the above numerical example. This diagram can also be used to determine the value of $b$ if $T_0$ is fixed due to some other considerations. Thus, if $c_r = 25$, and $T_0 = 15$, we need $b = 0.13$. If, however, $T_0 = 10$, $b$ must be 0.265.
Fig. 4.4  Contours of $T_0$ in the $b, c_3/(c_2 - c_1)$ plane ($a = 1348$).
4.4 OPTIMUM RELEASE TIME BASED ON COST CRITERION  
(MODEL OF SECTION 3)

4.4.1 Cost Model and Optimal Policy

The cost model for this case is similar to that given in Equation (4.5) and is (quantities are as defined in Section 4.3.1)

\[
c(T, t) = c_1 M(T) + c_2 [M(t) - M(T)] + c_3 T
\]

\[
(4.10)
\]

where

\[
M(x) = a(1 - e^{-bx^c})
\]

\[
(4.11)
\]

On differentiating (4.10) with respect to \( T \) and equating the result to zero, we get

\[
\wedge(T) = M'(T) = \frac{c_3}{c_2 - c_1}
\]

\[
(4.12)
\]

where

\[
\wedge(T) = a_T^{y-1} \cdot e^{-\beta T^y}
\]

\[
(4.13)
\]

\[\alpha = abc\]

\[\beta = b\]

\[\gamma = c\]
To solve (4.12) for $T$, we consider three cases depending on the value of $\gamma$, viz $\gamma > 1$, $\gamma = 1$, and $0 < \gamma < 1$.

**Case When $\gamma > 1$**

For this case, the failure distribution has an increasing failure rate followed by a decreasing failure rate. Then we see from (4.13) that $\Lambda(T)$ is a unimodal function of $T$ with $\Lambda(0) = \Lambda(\infty) = 0$. Also, its maximum $\Lambda(T_m)$ occurs at $T_m$ where

$$T_m = \frac{(\frac{\gamma-1}{\beta\gamma})^{1/\gamma}}{(\frac{\gamma-1}{\beta\gamma})^{1/\gamma}}$$

(4.14)

and

$$\Lambda(T_m) = a\frac{(\frac{\gamma-1}{\beta\gamma})^{(\gamma-1)/\gamma}}{c_3}.$$  

(4.15)

If $\Lambda(T_m) < \frac{c_3}{c_2 - c_1}$ and $\Lambda(T) < \frac{c_3}{(c_2 - c_1)}$ for $T \geq 0$, then it is easy to see that Equation (4.12) has no feasible solution for $T$. Therefore, for $T \geq 0$, $\frac{dC(T,t)}{dt} > 0$, and the minimum of $C(T;t)$ is at $T = 0$. In other words, if $\Lambda(T_m) < \frac{c_3}{c_2 - c_1}$ and $\Lambda(T) < \frac{c_3}{c_2 - c_1}$, then $T^* = 0$. This is shown graphically in Figure 4.5.
Fig. 4.5. Plots of $C(T;t)$ and $\Lambda(T)$ for $\gamma > 1$ and $\alpha(\frac{\gamma-1}{\beta \gamma})^\gamma < \frac{C_3}{C_2-C_1}$.
If \( \Lambda(T_m) = \frac{c_3}{(c_2 - c_1)} \), then \( \frac{d}{dT} C(T;t) > 0 \) for \( 0 < T < T_m \) and for \( T > T_m \). Then Equation (4.12) has a unique feasible solution \( T = T_m \).

However, \( T = T_m \) is an inflection point of \( C(T;t) \) for this case. Therefore, the minimum of \( C(T;t) \) is at \( T = 0 \), i.e., \( T^* = 0 \) as seen in Figure 4.6.

If, however, \( \Lambda(T_m) > \frac{c_3}{c_2 - c_1} \), there exist two feasible solutions \( T = T_1 \) and \( T = T_2 \), \( 0 < T_1 < T_2 < \infty \), which are the two positive roots of Equation (4.12). Also,

\[
\Lambda(T) < \frac{c_3}{c_2 - c_1}, \quad 0 < T < T_1, \ T > T_2
\]

and

\[
\Lambda(T) > \frac{c_3}{c_2 - c_1}, \quad T_1 < T < T_2.
\]

For this case, \( T_1 \) and \( T_2 \) can be obtained by solving Equation (4.12) numerically. It should also be pointed out that \( \frac{d}{dT} C(T;t) > 0 \) for \( 0 < T < T_1, \ T > T_2 \), and \( \frac{d}{dT} C(T;t) < 0 \) for \( T_1 < T < T_2 \). In this case, we consider the minimum of \( C(T;t) \) for the following three cases (see Figure 4.7).
Fig. 4.6. Plots of $C(T;t)$ and $\Lambda(T)$ for $\gamma > 1$ and $\alpha(\frac{\gamma-1}{\beta^2/e}) = \frac{C_3}{C_2 - C_1}$. 
Fig. 4.7. Plots of $C(T;t)$ and $\Lambda(T)$ for $\gamma > 1$ and $a(\frac{\gamma-1}{\beta y}) \gamma > \frac{c_3}{c_2-c_1}$.
Case A. If $C(0; t) > C(T_2; t)$, then the minimum of $C(T; t)$ is at $T = T_2$ for $t \geq T_2$ and is at $T = t$ for $t < T_2$.

Case B. If $C(0; t) < C(T_2; t)$, then the minimum of $C(T; t)$ is at $T = 0$.

Case C. If $C(0; t) = C(T_2; t)$, then the minimum of $C(T; t)$ is at $0$ or $T_2$ for $t \geq T_2$ and is at $T = 0$ for $t < T_2$.

Case When $\gamma = 1$

For this case, $\Lambda(T) = u e^{-\beta T}$ where $\alpha = ab$ and $\beta = b$.

Now $\Lambda(T)$ is a monotonically decreasing function of $T$ and $\Lambda(0) = u$. If $\alpha \leq \frac{c_3}{c_2 - c_1}$, (4.12) has no feasible solution and $\frac{dC(T; t)}{dT} > 0$ for $T \geq 0$. Therefore, the minimum of $C(T; t)$ is at $T = 0$, i.e., $T^* = 0$. If, however, $\alpha > \frac{c_3}{c_2 - c_1}$, then there exists a unique solution of (4.12) given by

$$T_0 = \frac{1}{\beta} \cdot \ln\left(\frac{\alpha (c_2 - c_1)}{c_3}\right).$$

(4.16)

From the fact that $\frac{dC(T; t)}{dT} > 0$ for $0 \leq T < T_0$ and $\frac{dC(T; t)}{dT} > 0$ for $T > T_0$, the minimum of $C(T; t)$ is at $T = T_0$ for $t \geq T_0$ and at $T = t$ for $t < T_0$. 

4-21
Case When $0 < \gamma < 1$

For this case, $\Lambda(T)$ is a monotonically decreasing function of $T$ with $\Lambda(0) = \infty$ and $\Lambda(\infty) = 0$. Then the unique positive root of Equation (4.12) is the solution. Further, since $\frac{dC(T;t)}{dT} < 0$ for $0 < T < T_3$ and $\frac{dC(T;t)}{dT} > 0$ for $T > T_3$, the minimum of $C(T;t)$ is at $T = T_3$ for $t < T_3$ and at $T = t$ for $t < T_3$ as seen in Figure 4.8. We can summarize the above results in the following theorem.

**Theorem 4.2.** Suppose that $\alpha$, $\beta$, $\gamma$, $c_1$, $c_2$, and $c_3$ are all greater than zero. Then the optimum release time $T^*$ is given by the following expressions for the cases when $\gamma > 1$, $\gamma = 1$, and $0 < \gamma < 1$.

**Case When $\gamma > 1$**

**Case A:** If $a(\gamma - 1)(\gamma - 1)/\gamma < \frac{c_3}{c_2 - c_1}$, then $T^* = 0$.

**Case B:** If $a(\gamma - 1)(\gamma - 1)/\gamma > \frac{c_3}{c_2 - c_1}$, then there exist two feasible solutions $T = T_1$ and $T = T_2$ ($0 < T_1 < T_2 < \infty$) which are the two positive roots of Equation (4.12) and the optimum release time is as follows:
\[ C(0; t) = \frac{C_3}{C_2 - C_1} \]

**Fig. 4.8.** Plots of \( C(T; t) \) and \( \Lambda(T) \) for \( 0 < \gamma < 1 \).
If $C(0; t) > C(T_2; t)$, then $T^* = \min\{T_2; t\}$

If $C(0; t) < C(T_2; t)$, then $T^* = 0$

If $C(0; t) = C(T_2; t)$, then $T^* = 0$ and $T_2, \ t > T_2$

and $T^* = 0$ for $t < T_2$

Case When $\gamma = 1$

If $\alpha \leq \frac{c_3}{c_2 - c_1}$, then $T^* = 0$. If $\alpha > \frac{c_3}{c_2 - c_1}$, then there exists a unique solution

$$T_0 = \frac{1}{\beta} \ln\left(\frac{\alpha (c_2 - c_1)}{c_3}\right)$$

of Equation (4.12) and the optimum release time is $T^* = \min(T_0; t)$.

Case When $0 < \gamma < 1$

For this case, Equation (4.12) has a unique positive root $T_3$ which is the solution, and the optimum release time is $T^* = \min(T_3; t)$.
5. BIBLIOGRAPHY


A1. INTRODUCTION

In this section we describe the development of stochastic models for performance and cost evaluation of hardware-software systems in the operational phase.

Section A.2 deals with the development of stochastic models for system performance assessment. The state of the system is described by up or down states of the hardware and the software system and by the number of errors in the software system. The hardware-software system is down if either the hardware or the software system is down, and up if both are up. The hardware failure distribution is exponential with failure rate $\beta$. The software failure distribution between occurrences of software failures is also exponential with a failure rate $i\lambda$, where $i = 0,1,\ldots,N$ is the number of remaining errors in the system. The repair rates are exponential with parameters $\gamma$ and $\mu$, and the probabilities of imperfect repair are $p_h$ and $p_s$ for the hardware and the software systems, respectively.

Based on this model, expressions for various stochastic performance measures are also developed in Section A2. These are distribution of time to a specified number of remaining software errors; state
occupancy probabilities; expected number of hardware, software, and hardware-software failures detected by time \( t \); system reliability, availability and average availability.

In some cases, an improvement in one performance measure causes a worsening of another. For example, an improved system availability causes an increase in the expected number of failures. In order to evaluate the effect of these conflicting measures on system performance, cost models are developed in Section A3 for the hardware, software, and hardware-software systems. Each model gives expected total cost by time \( t \) and consists of three cost elements; the cost of failures, the cost of repairs, and the cost due to system unavailability. The results of a numerical study to investigate the effects of cost factors, failure rates, and repair rates on the expected number of failures, average availability and expected total cost/unit time are also discussed.
A2. A MARKOV MODEL FOR HARDWARE-SOFTWARE SYSTEM AND PERFORMANCE MEASURES

In this section we develop a stochastic model and expressions for the performance measures of a hardware-software system. The basic model is developed in Section A2.1 and assumes the system behavior to be Markovian.

In order to use this model to evaluate and predict the system performance, we generally need expressions for the appropriate quantitative measures. Such expressions for the following measures are derived in Sections A2.2 to A2.5.

(i) Distribution of time to a specified number of remaining errors in the software system.
(ii) State occupancy probabilities.
(iii) System reliability and availability.
(iv) Expected number of software, hardware, and total failures by time t.

A2.1. System Description and Model Development

Consider a system consisting of hardware and software components, all of which are subject to random failures. The hardware components fail due to either defects or wear-out.
A software component is said to fail when a fault, a specific manifestation of an error in the program, is evoked by some input data resulting in the program not correctly computing the required function. Whenever any of these failures occurs, the system goes out of operation. A repair activity is then undertaken to remove the cause of the failure and bring the system back to an operational state.

In the present study, we assume that the hardware and software components can be viewed as a single system each. In other words, the hardware-software system will be treated as 2-unit (or 2-system) systems, one representing the hardware components and the other the software components. The up and down states of such a system are shown in Figure A2.1.

We develop a model for the stochastic behavior of the system under the following assumptions:

(i) The errors in the software system are independent of each other and each has an error occurrence rate \( \lambda \).

(ii) The failures of the hardware system are independent of each other and have a constant occurrence rate \( \beta \). Only those failures which cause the system to go down are considered.

(iii) The probability of two or more software or hardware failures occurring simultaneously
is negligible.

(iv) The time to remove a software error, when there are \( i \) errors in the system, follows an exponential distribution with parameter \( \mu_i \).

(v) The time to remove the cause of a hardware failure follows an exponential distribution with parameter \( \gamma \).

(vi) Failures and repairs of the hardware system are independent of both the failures and repairs of the software system.

(vii) At most one software error is removed at correction time and no new software errors are introduced during the error removal (correction) phase.

(viii) When the system is inoperative due to the occurrence of a software failure, the error causing the failure, when detected, is corrected with probability \( p_s (0 \leq p_s \leq 1) \), while with probability \( q_s (p_s + q_s = 1) \) the error is not removed. Thus, \( q_s \) is the probability of imperfect maintenance of software.

(ix) After the occurrence of a hardware failure, the cause of the failure is removed with probability \( p_h (0 \leq p_h \leq 1) \) while with probability \( c_h (p_h + c_h = 1) \), the cause is not removed. Thus, \( c_h \) is
the probability of imperfect maintenance of hardware.

(x) The system is considered to be inoperative whenever it is under maintenance following a hardware or a software failure.

Now, we examine the failure and repair times of the software and hardware systems independently, based upon the above assumptions.

Software failures, from assumptions (i) and (iii), follow an exponential distribution. Let $i$ be the number of errors in the software system. Then the probability density function (pdf) of the time to next software failure, $T_i$, is given by the distribution of the first order statistic of $i$ exponential distributions each with parameter $\lambda$, i.e.

$$ f_i(t) = \binom{i}{1}(\lambda e^{-\lambda t})(e^{-\lambda t})^{i-1} $$

or

$$ f_i(t) = i\lambda e^{-i\lambda t} \quad \text{(A2.1)} $$

Letting $\lambda_i = i\lambda$, the pdf and the cumulative distribution function (cdf) of $T_i$ can be written as

$$ f_i(t) = \lambda_i e^{-\lambda_i t} \quad \text{(A2.2)} $$

and

$$ F_i(t) = 1 - e^{-\lambda_i t} \quad \text{(A2.3)} $$

From assumption (iv), the cdf of the software maintenance
time when there are \( i \) errors in the system, \( W_i \), is

\[
P(W_i \leq t) = 1 - e^{-\nu_i t} \tag{A2.4}
\]

The cdf's of the hardware time to failure, \( U \), and maintenance time, \( V \), from assumptions (ii) and (v), respectively, are:

\[
P(U \leq t) = 1 - e^{-\beta t} \tag{A2.5}
\]

and

\[
P(V \leq t) = 1 - e^{-\gamma t}. \tag{A2.6}
\]

To summarize, hardware failures and repairs occur according to exponential distributions with parameters \( \beta \) and \( \gamma \), respectively. These parameters are considered to remain constant. The distribution of the times between software failures also follows an exponential distribution, but its parameter, \( \lambda_i \), also varies with the number of errors remaining in the software system, \( i \). The distribution of the maintenance time for software is again exponential with a parameter \( \mu_i \) which changes with \( i \).

Now, we consider the failure phenomenon in the total hardware-software system. Suppose there are \( i \) errors in the software and the total system is operational. Let

\[
Y_i = \min(T_i, U) \tag{A2.7}
\]

It can be easily shown that \( Y_i \) has an exponential distribution with parameter \( (\beta + \lambda_i) \) and

\[
F_{Y_i}(y) = 1 - e^{-(\beta + \lambda_i)y} \tag{A2.8}
\]

The probability that a software failure will occur before a hardware failure is
\[ P(T_i < U) = \int_0^\infty P(U > t_i | T_i = t_i) \cdot dF_i(t) \]

\[ = \int_0^\infty P(U > t) \cdot \lambda_i e^{-\lambda_i t} \cdot dt \]

\[ = \int_0^\infty \lambda_i e^{-(\beta + \lambda_i) t} dt \]

or

\[ P(T_i < U) = p_i = \frac{\lambda_i}{\beta + \lambda_i}, \quad i = 0, 1, \ldots, N \tag{A2.9} \]

Similarly, the probability that a hardware failure occurs before a software failure is

\[ P(U < T_i) = q_i = \frac{\beta}{\beta + \lambda_i}, \quad i = 0, 1, \ldots, N \tag{A2.10} \]

In other words, when the hardware-software system is operational with \( i \) software errors, the time to next failure is given by \( Y_i \). The probability of the next failure being a software failure is \( p_i \) and being a hardware failure is \( q_i \).

Let \( X(t) \) denote the state of the system at time \( t \); where

\[
X(t) = \begin{cases} 
  i, & \text{the system is operational while there are } i \text{ errors remaining in the software system, } i = 0, 1, 2, \ldots, N. \\
  i_s, & \text{the system is down for maintenance of software with } i \text{ software errors, } i_s = 1_s, 2_s, \ldots, N_s \\
  i_h, & \text{the system is down for maintenance of hardware with } i \text{ software errors, } i_h = 0_h, 1_h, \ldots, N_h
\end{cases} \tag{A2.11}
\]
The transitions between the states of the system, i.e., in $X(t)$, are governed by assumptions (i) through (x) and Equations (A2.9) and (A2.10). The transition probability matrix for the $X(t)$ process is given in Equation (A2.12) and a diagrammatic representation of such transitions is given in Figure A2.2.

\[
P_{k,j} = \begin{bmatrix}
N & N_h & N_s & N-1 & N-1_h & N-1_s & N-2 & \ldots & 1 & l_h & l_s & 0 & 0_h \\
N & 0 & q_N & P_N & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
N_h & p_h & q_h & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
N_s & q_s & 0 & 0 & P_s & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
N-1 & 0 & 0 & 0 & 0 & q_{N-1} & P_{N-1} & 0 & \ldots & 0 & 0 & 0 & 0 \\
N-1_h & 0 & 0 & 0 & p_h & q_h & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
N-1_s & 0 & 0 & 0 & q_s & 0 & 0 & P_s & \ldots & 0 & 0 & 0 & 0 \\
N-2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & q_1 & P_1 & 0 \\
l_h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & p_h & q_h & 0 & 0 \\
l_s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & q_s & 0 & 0 & p_s \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & 0 \\
0_h & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \ldots & 0 & 0 & 0 & P_h & q_h
\end{bmatrix}
\] (A2.12)
Figure A2.2. Diagrammatic Representation of Transitions Between States of X(t)
To summarize the system behavior, consider once again the above situation, i.e., the total system is operational with $i$ software errors. The time to a failure is governed by $Y_i$. If a software failure occurs, and the probability of this occurring is $p_i$, the system goes into a down state $i_s$. The system undergoes software maintenance and, after a random time governed by Equation (A2.4), goes to state $i$ with probability $q_s$ and to state $(i-1)$ with probability $p_s$.

If the failure is a hardware failure, and the probability of this happening is $q_i$, the system goes into a down state, $i_h$. Following a repair for the failure according to Equation (A2.5), the system goes back to state $i$ with probability $p_h$ or stays in state $i_h$ with probability $q_h$.

The above system behavior is valid only until the software is error-free. After the software is error-free, the total system reduces to a hardware system only.

Thus, we see that the stochastic process $X(t)$ forms a semi-Markov process. It makes transitions as described above and the times spent in various states are random, given by $Y_i$, $W_i$, or $V$, depending on the state. A typical realization of the $X(t)$ process corresponding to Figure A2.2 is shown in Figure A2.3.

Let $Q_{k,j}(t)$, $k,j = i, i_s, i_h$, be the one step transition probability that after making a transition into state $k$, the process $X(t)$ next makes a transition into state $j$ in an amount of time less than or equal to $t$. Then, $Q_{k,j}(t)$

A-12
Figure A2.3. A Typical Realization of $X(t)$ Process.
is given by the product of $P_{k,j}$ and the cdf up to $t$ of the time corresponding to state $k$. Thus, for $k = i$, and $j = i_s$, we have

$$Q_{i,i_s}(t) = P_{i,i_s} \cdot F_{X_i}(t) \quad (A2.13)$$

The expressions for various $Q_{k,j}$'s are as follows:

$$Q_{i,i_s}(t) = \frac{\lambda_i}{\beta + \lambda_i} (1 - e^{-(\beta+\lambda_i)t})$$

$$Q_{i,i_h}(t) = \frac{\beta}{\beta + \lambda_i} (1 - e^{-(\beta+\lambda_i)t})$$

$$Q_{i_s,i}(t) = q_s (1 - e^{-\mu_i t})$$

$$Q_{i_s,i-1}(t) = p_s (1 - e^{-\mu_i t})$$

$$Q_{i_h,i}(t) = p_h (1 - e^{-\gamma t})$$

$$Q_{i_h,i_h}(t) = q_h (1 - e^{-\gamma t})$$

The expressions for $Q_{k,j}(t)$'s given by Equation (A2.14) constitute the basic equations that describe the stochastic behavior of the $X(t)$ process. These equations will be used in the subsequent sections to derive the system performance measures. We will need the Laplace-Stieltjes transforms of the $Q_{k,j}(t)$'s and some related results. These are given below.
Let $\mathcal{L}$ and $\mathcal{LS}$ denote the Laplace and Laplace Stieltjes transform, respectively; and for any function $g$ and $G$, let

$$g^*(s) = \mathcal{L}(g(t)), \quad \text{and} \quad \tilde{G}(s) = \mathcal{LS}(G(t)).$$

The Laplace-Stieltjes transform of the above $Q_{i,j}(t)$'s are

$$\mathcal{LS}(Q_{i,i}) = \frac{\lambda_i}{s + \beta + \lambda_i}, \quad (A2.16)$$

$$\mathcal{LS}(Q_{i,i}) = \frac{\beta}{s + \beta + \lambda_i}, \quad (A2.17)$$

$$\mathcal{LS}(Q_{i,i}) = \frac{q_i}{s + \gamma_i}, \quad (A2.18)$$

$$\mathcal{LS}(Q_{i,i}) = \frac{p_{i} \gamma}{s + \gamma}, \quad (A2.19)$$

$$\mathcal{LS}(Q_{i,i}) = \frac{q_{i} \gamma}{s + \gamma}. \quad (A2.20)$$

The following Lemmas from the basic Laplace, Laplace-Stieltjes transforms and their inverses will be useful for our analysis (see Abramowitz et al., 1965, and Muth, 1977).

**Lemma A2.1.** (Linearity property). If

$$h(t) = Af(t) + Bg(t)$$

then

$$h^*(s) = \mathcal{L}(h(t)) = Af^*(s) + Bg^*(s).$$
Lemma A2.2. The Laplace transform of pdf \( f(t) \) is equivalent to the Laplace-Stieltjes transform of its cdf \( F(t) \).

\[
f^*(s) = \mathcal{L}\{f(t)\} = \int_0^\infty e^{-st}f(t)dt
\]

\[
= \int_0^\infty e^{-st}dF(t)
\]

\[
= \mathcal{L}S\{F(t)\}.
\]

Lemma A2.3. (Heaviside Expansion Theorem). If

(i) \( q(s) = (s-a_1)(s-a_2) \cdots (s-a_m) \),

where \( a_1 \neq a_2 \neq \ldots \neq a_m \),

(ii) \( p(s) \) is a polynomial of degree \( m \), and

(iii) \( f^*(s) = \frac{p(s)}{q(s)} \),

then

\[
f(t) = \sum_{n=1}^m \frac{p(a_n)}{q'(a_n)} a_n t.
\]

where \( q'(a_i) = \sum_{i,j=1}^{m} (a_i - a_j) \),

and

\[
F(t) = \sum_{n=1}^m \frac{p(a_n)}{q'(a_n)} \cdot \frac{1}{a_n} (e^{a_n t} - 1).
\]

If one of the \( a_n \), i.e. \( a_i = 0, 1 \leq i \leq m \), then

\[
F(t) = \frac{p(a_i)}{q'(a_i)} t + \sum_{n=1}^m \frac{p(a_n)}{q'(a_n)} \frac{1}{a_n} (e^{a_n t} - 1).
\]
A2.2 Distribution of Time to a Specified Number of Remaining Errors in a Software System

The errors remaining in the software system are sources of failures and we would like to remove them as soon as possible. However, it is not always feasible and/or practical to remove all of them in a reasonable time. In that case, we would like to know the distribution of time to $n$ ($0 \leq n \leq N$) remaining errors.

Let $T_{i,n}$ be the first passage time for operational state $i$ to operational state $n$ and let $G_{i,n}$ be its cdf. Now we derive the equations for $g_{N,n}(t)$ and $G_{N,n}(t)$, the pdf and cdf, respectively, of $T_{N,n}$.

A2.2.1 Distribution of $T_{N,n}$

Consider a time interval $(r,r+dr)$. For any $i$, the probability of going from $i$ to $i_s$ in this interval is $dQ_{i,i_s}(r)$ and the probability of going from $i$ to $i_h$ is $dQ_{i,i_h}(r)$. Once the $X(t)$ process reaches either $i_s$ or $i_h$, further transitions in it will be governed by cdf's, $G_{i_s,n}$ and $G_{i_h,n}$, respectively. Thus, the renewal equation for $G_{i,n}$, $i = n+1, \ldots, N$ can be written as
\[ G_{i,n}(t) = \sum_{k \in E} \int_{0}^{t} G_{k,n}(t-x) dQ_{i,k}(x) \]

\[ = Q_{i,i} * G_{i,n}(t) + Q_{i,i} * G_{i,n}(t) \]

\[ = Q_{i,i} * G_{i,n}(t) + Q_{i,i} * G_{i,n}(t) \]

\[ + Q_{i,i} * Q_{i,n,i-1} * G_{i-1,n}(t), \quad (A2.22) \]

where \( E \) is the state space, \( G_{n,n} = 1 \), \( Q_{H} = \sum_{j}^{j} Q_{i h,j} \), and \( Q_{i h,j} \) is the \( j \)-fold convolution of \( Q_{i h,i} \) with itself.

Taking the Laplace-Stieltjes (L-S) transform of Equation (A2.22) we get

\[ \tilde{G}_{i,n}(s) = \tilde{Q}_{i,i}(s) \tilde{Q}_{H}(s) \tilde{Q}_{i,i}(s) \tilde{G}_{i,n}(s) \]

\[ + \tilde{Q}_{i,i}(s) \tilde{Q}_{i,i}(s) \tilde{G}_{i,n}(s) \]

\[ + \tilde{Q}_{i,i}(s) \tilde{Q}_{i,i}(s) \tilde{G}_{i,n}(s), \quad (A2.23) \]

where

\[ \tilde{Q}_{H}(s) = \sum_{j=0}^{\infty} \tilde{Q}_{i h,j}(s) = \frac{s + \gamma}{s + p_{h} \gamma} \]

and, from Equations (A2.16) to (A2.21),

\[ \tilde{Q}_{i,i}(s) = \frac{s}{s + \beta + \lambda \gamma} \]

\[ \tilde{Q}_{i h,i}(s) = \frac{\gamma}{s + \gamma} \]

\[ \tilde{Q}_{i h,i}(s) = \frac{\gamma}{s + \gamma} \]
\[ \tilde{Q}_{i,s} (s) = \frac{\lambda_i}{s + \beta + \lambda_i}, \]
\[ \tilde{Q}_{i,s'} (s) = \frac{q_i u_i}{s + u_i}, \]
\[ \tilde{Q}_{i,s'} (s) = \frac{p_i u_i}{s + u_i}. \]

On substituting the expressions for the various L-S transforms in Equation (A2.23) and simplifying, we get
\[ \tilde{g}_{i,n} (s) = a_i \tilde{g}_{i,n} (s) + b_i \tilde{g}_{i-1,n} (s), \quad (A2.24) \]
where
\[ a_i = \frac{p_i \gamma (s+\mu_i) + q_i \lambda_i u_i (s+p_i \gamma)}{(s+p_i \gamma) (s+\beta+\lambda_i) (s+\mu_i)}, \quad (A2.25) \]
\[ b_i = \frac{p_i \lambda_i u_i}{(s+\mu_i) (s+\beta+\lambda_i)}. \quad (A2.26) \]

For \( i = n+1, \tilde{g}_{i-1,n} (s) = \tilde{g}_{n,n} (s) = 1, \) and
\[ \tilde{g}_{i,n} (s) = \tilde{g}_{n+1,n} (s) = \frac{b_i}{1 - a_i} \quad (A2.27) \]
\[ = \frac{p_i \lambda_i u_i (s+p_i \gamma)}{(s+x_{1,i}) (s+x_{2,i}) (s+x_{3,i})}, \quad (A2.28) \]

where \(-x_{1,i}, -x_{2,i}, \) and \(-x_{3,i}\) are the roots of the polynomial
\[ s^3 + s^2 (\lambda_i + \mu_i + \beta + p_i \gamma) + s (p_i \lambda_i u_i + \beta u_i + \lambda_i p_i \gamma + \mu_i p_i \gamma) + p_i p_i \gamma \lambda_i u_i \]
\[ \tilde{g}_{n,n} (s) = \prod_{i=n+1}^{N} \frac{p_i \lambda_i u_i ((s+p_i \gamma))}{(s+x_{1,i}) (s+x_{2,i}) (s+x_{3,i})} \quad (A2.29) \]

Further, let

A-19
\[ x_{1,n+1} = x_1 \]
\[ x_{2,n+1} = x_2 \]
\[ x_{3,n+1} = x_3 \]
\[ x_{1,n+2} = x_4 \]
\[ x_{2,n+2} = x_5 \]
\[ x_{3,n+2} = x_6 \]
\[ x_{1,N} = x_1(N-n) \]
\[ x_{2,N} = x_2(N-n) \]
\[ \vdots \]
\[ x_{3,N} = x_3(N-n) \]

and

\[ K = 3(N-n) \]

\[ g_{N,n}(t) = \frac{\left\{ p_s(s+p_h\gamma)^{N-n} \right\}^N \prod_{i=n+1}^{n} \lambda_i \mu_i }{\prod_{j=1}^{K} (s+x_j)} \] \hspace{1cm} (A2.31)

By using the results from Lemma A2.3, the pdf and the cdf of \( T_{N,n} \) are obtained from Equation (A2.31) as

\[ g_{N,n}(t) = \frac{\left\{ p_s(s+p_h\gamma)^{N-n} \right\}^N \prod_{i=n+1}^{n} \lambda_i \mu_i }{\prod_{j=1}^{K} (s+x_j)} e^{-x_j t} \] \hspace{1cm} (A2.32)

and

\[ G_{N,n}(t) = \frac{\left\{ p_s(s+p_h\gamma)^{N-n} \right\}^N \prod_{i=n+1}^{n} \lambda_i \mu_i }{\prod_{j=1}^{K} (s+x_j)} \cdot \frac{1}{-x_j} (e^{-x_j t} - 1) \] \hspace{1cm} (A2.33)
where
\[ U_{n+1}^N = \prod_{i=n+1}^{N} (p_s \lambda_i \nu_i). \]

The distribution function of the first passage
time to enter a state corresponding to a specified number
of remaining software errors will be useful in the study
and analysis of the other performance measures.

A2.2.2 Mean and Variance of \( T_{N,n} \)

Now,
\[ E[T_{N,n}] = \int_0^\infty g_{N,n}(t) dt \]  \hspace{1cm} (A2.34)

Substituting for \( g_{N,n}(t) \) from Equation (A2.32), we get
\[ E[T_{N,n}] = \sum_{j=1}^{K} \frac{U_{n+1}^N (-x_j + p_h \gamma)^{N-n}}{K} \prod_{i=1}^{K} (-x_j + x_i) \int_0^\infty t e^{-x_j t} dt \]

or
\[ E[T_{N,n}] = \sum_{j=1}^{K} \frac{U_{n+1}^N (-x_j + p_h \gamma)^{N-n}}{K} \prod_{i=1}^{K} (-x_j + x_i) \cdot \frac{1}{(x_j)^2} \]  \hspace{1cm} (A2.35)
Similarly, to get the variance of $T_{N,n}$ we have

$$E[T_{N,n}^2] = \sum_{j=1}^{K} \frac{U_{n+1}^{N}(-x_j + P_h \gamma)^{N-n}}{\prod_{i=1, i \neq j}^{K} (-x_j + x_i)} \cdot \frac{2}{(x_j)^3}$$

(A2.36)

and

$$\text{Var}[T_{N,n}] = E[T_{N,n}^2] - E^2[T_{N,n}].$$

(A2.37)

A2.2.2 Illustrative Example

Consider a system with $N = 10$ errors, $p_s = 0.9$, and $P_h = 0.9$. Assume that $\lambda_i = i\lambda$, $\mu_i = i\mu$, and the parametric values are $\lambda = .02$, $\mu = .05$, $\beta = .01$, and $\gamma = .025$. We are interested in the distribution of $T_{N,n}$, $n = 0,1,2,...,8,9$. The pdf's and cdf's of $T_{N,n}$ for various values of $n$ and for $t$ from 0 to 500 units are computed from Equation (A2.32) and (A2.33), respectively, and are shown in Figures A2.4 and A2.5, respectively. Also, the means and variances of these distributions are obtained from Equations (A2.35) and (A2.37) respectively, and are summarized in Table A2.1. From Figures (A2.4) and (A2.5), we notice that the distributions are highly dependent on $n$. Also, as expected, the distribution of the time to an error-free software system has a large mean and a large variance (see Table A2.1). The mean and variance of $T_{10,n}$ for $P_h = 1.0$ are also given in Table A2.1. We note that both of these values are smaller.
\[ \lambda_i = i\lambda \quad \lambda = 0.02 \quad \beta = 0.01 \quad p_s = 0.9 \]
\[ \mu_i = i\mu \quad \mu = 0.05 \quad \gamma = 0.025 \quad p_h = 0.9 \]
\[ N = 10, \quad n = 9, 8, \ldots, 1, 0 \]

Figure A2.4. Probability Distribution Function of First Passage Time to n.
Figure A2.5. Cumulative Distribution Function of First Passage Time to n.

\[ \lambda = .02 \quad \beta = .01 \quad p_s = .9 \]
\[ u = .05 \quad \gamma = .025 \quad p_h = .9 \]
\[ N = 10, \quad n = 9, 8, 7, \ldots, 2, 1, 0 \]

A-24
<table>
<thead>
<tr>
<th>TO STATE</th>
<th>FH = 0.9</th>
<th></th>
<th>FH = 1.0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEAN</td>
<td>VARIANCE</td>
<td>S.D.</td>
<td>MEAN</td>
</tr>
<tr>
<td>9</td>
<td>10.2</td>
<td>292.4</td>
<td>17.1</td>
<td>10.0</td>
</tr>
<tr>
<td>8</td>
<td>21.6</td>
<td>626.2</td>
<td>25.0</td>
<td>21.1</td>
</tr>
<tr>
<td>7</td>
<td>34.4</td>
<td>1014.5</td>
<td>31.9</td>
<td>33.6</td>
</tr>
<tr>
<td>6</td>
<td>49.1</td>
<td>1476.8</td>
<td>38.4</td>
<td>47.9</td>
</tr>
<tr>
<td>5</td>
<td>66.2</td>
<td>2045.1</td>
<td>45.1</td>
<td>64.7</td>
</tr>
<tr>
<td>4</td>
<td>86.7</td>
<td>2775.7</td>
<td>52.7</td>
<td>84.6</td>
</tr>
<tr>
<td>3</td>
<td>112.3</td>
<td>3780.0</td>
<td>61.5</td>
<td>109.6</td>
</tr>
<tr>
<td>2</td>
<td>146.4</td>
<td>5321.6</td>
<td>72.9</td>
<td>142.9</td>
</tr>
<tr>
<td>1</td>
<td>197.7</td>
<td>8241.5</td>
<td>90.8</td>
<td>192.9</td>
</tr>
<tr>
<td>0</td>
<td>300.1</td>
<td>17726.3</td>
<td>133.1</td>
<td>292.9</td>
</tr>
</tbody>
</table>
than those for $p_n = 0.9$ because of an improvement in the hardware system maintenance activity.

A2.3 State Occupancy Probabilities

In this Section we are interested in deriving expressions for the probability that the system is operational at time $t$ with a specified number of remaining software errors. Let $P_{N,n}(t)$ be the probability that the system is operational at time $t$ with $n$ remaining software errors, given that it was in operation at time $t = 0$ with $N$ software errors, i.e.

$$P_{N,n}(t) = P\{X(t)=n|X(0)=N\}, \quad n = 0, 1, \ldots, N \quad (A2.38)$$

We call $P_{N,n}(t)$ the (operational) state occupancy probability. By conditioning on the first up-down cycle of the process and using an approach similar to that of Section A2.2 we get the following renewal equation for $P_{n,n}(t)$

$$P_{n,n}(t) = e^{-(\lambda + \beta)t} + Q_{n,n}P_{n,n}(t) \quad (A2.39)$$

By conditioning on the first passage time, we get

$$P_{N,n}(t) = P_{n,n}G_{N,n}(t). \quad (A2.40)$$

To obtain the L-S transform of $P_{N,n}(t)$, we take the L-S transforms of Equations (A2.39) and (A2.40) and solve the resulting equations. Let $a_1$, $b_1$, $s_i$, and $x_i$ be as given in Equations (A2.25), (A2.26), and (A2.30), respectively.
Then, by having \( G_{N,N}(s) = 1 \), and letting
\[
A(s) = s\beta(s + \nu_n) + \lambda_n(s + p_s u_n)(s + p_h Y),
\]
\[
B(s) = (s - x_1, n)(s - x_2, n)(s - x_3, n),
\]
the L-S transforms of \( P_{N,n}(t) \) and \( P_{N,0}(t) \), respectively, are
\[
P_{N,n}(s) = (1 - \frac{A(s)}{B(s)}) G_{N,n}(s) \tag{A2.41}
\]
and
\[
P_{N,0}(s) = (1 - \frac{\beta}{s + \beta + p_h Y}) G_{N,0}(s)
\]

The expressions for \( P_{N,0}(t) \) and \( P_{N,n}(t) \), \( n = 1,\ldots,N \)
are obtained from the results of Lemma A2.1 to A2.3 as

\[
P_{N,n}(t) = G_{N,n}(t) \sum_{j=1}^{K} \frac{U_{n+1} (-x_j + p_h Y)^{N-n-1} A(-x_i) \cdot (1 - e^{-x_j t})}{\prod_{i=1, i\neq j}^{K} (-x_j + x_i)}
\]
and

\[
P_{N,0}(t) = G_{N,0}(t) \sum_{j=1}^{K_1} \frac{U_{n+1} (-x_j + p_h Y)^N \cdot (1 - e^{-x_j t})}{\prod_{i=1, i\neq j}^{K_1} (-x_j + x_i)}
\]

where \( K = 3(N-n+1) \) and \( K_1 = 3N+1 \) are the number of roots in the denominator.

A2.4 System Reliability and Availability

A2.4.1 System Reliability

The reliability of a system at time \( x \) is given by
\[
\bar{F}(x) = 1 - F(x)
\]
where \( F \) is the life distribution of the system. The corresponding conditional reliability of a unit of age \( t \) is
\[ \hat{F}(x|t) = \frac{\hat{F}(t + x)}{F(t)} \text{, if } \hat{F}(t) > 0 \]

Consider our hardware-software system. At \( t = 0 \), the initial number of software errors in the system is equal to \( N \). The reliability of the system at this stage is

\[
P \{\text{up time} > x\} = P \{\min(U, T_N) > x\} = P\{U > x\} \cdot P\{T > x\} = e^{-(\beta + \lambda_N)x}
\]

Next, consider some time \( t > 0 \) when the system has just been repaired and there are \( i \) remaining errors. The reliability of the system is

\[
P \{\text{up time} > x|X(t) = i\} = P \{\min(U, T_i) > x\} = e^{-(\beta + \lambda_i)x}
\]

A2.4.2 System Availability

Another useful measure of system performance is its availability, which is defined as the probability that it is operational at some given time \( t \). In our case, the system will be operational if the hardware system is in an up state and the software system is in an up state with \( n \) remaining errors, \( n = 0, 1, \ldots, N \). In Section A2.3, we derive the expressions for \( P_{N,n}(t) \), the probability that the system is operational at time \( t \) with \( n \) errors in the software system, given that it was operational at \( t = 0 \) with \( N \) software errors. Thus, the system availability can be defined as

\[
A(t) = \sum_{n=0}^{N} P_{N,n}(t)
\]

To see the behavior of \( A(t) \) we consider an example with \( N = 10 \), \( p_s = 0.9 \), \( p_h = 0.9 \), \( \lambda = 0.02 \), \( \mu = 0.05 \), \( \beta = 0.01 \) and A-28
Figure A2.6. State Occupancy Probabilities and System Availability.

\[ \lambda = 0.02 \quad \beta = 0.01 \quad \pi_s = 0.9 \]

\[ \mu = 0.05 \quad \gamma = 0.025 \quad \pi_h = 0.9 \]

\[ N = 10, \quad n = 10, 9, \ldots, 1, 0 \]
Table A2.2
Selected Values of $P_{N,n}(t)$ and $A(t)$
$N = 10$

<table>
<thead>
<tr>
<th>n</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>400</th>
<th>500</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>.003</td>
<td>.001</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>9</td>
<td>.009</td>
<td>.003</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>8</td>
<td>.030</td>
<td>.006</td>
<td>.001</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>7</td>
<td>.081</td>
<td>.012</td>
<td>.002</td>
<td>.000</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>6</td>
<td>.149</td>
<td>.024</td>
<td>.004</td>
<td>.001</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>5</td>
<td>.166</td>
<td>.052</td>
<td>.008</td>
<td>.001</td>
<td>.000</td>
<td>.000</td>
</tr>
<tr>
<td>4</td>
<td>.106</td>
<td>.106</td>
<td>.018</td>
<td>.004</td>
<td>.001</td>
<td>.000</td>
</tr>
<tr>
<td>3</td>
<td>.036</td>
<td>.159</td>
<td>.042</td>
<td>.011</td>
<td>.003</td>
<td>.001</td>
</tr>
<tr>
<td>2</td>
<td>.006</td>
<td>.139</td>
<td>.105</td>
<td>.037</td>
<td>.011</td>
<td>.003</td>
</tr>
<tr>
<td>1</td>
<td>.000</td>
<td>.054</td>
<td>.215</td>
<td>.146</td>
<td>.074</td>
<td>.033</td>
</tr>
<tr>
<td>0</td>
<td>.000</td>
<td>.005</td>
<td>.191</td>
<td>.437</td>
<td>.578</td>
<td>.645</td>
</tr>
</tbody>
</table>

$A(t)$ | .586 | .561 | .586 | .637 | .667 | .682|

A-30
\( \gamma = 0.025 \). For these values, distributions \( P_{N,n}(t) \), \( n = 0, 1, \ldots, 10 \), \( t \) from 0 to 500 are obtained as described in Section A2.3. Selected values of \( P_{N,n}(t) \) are given in Table A2.2 and the probability distributions are plotted in Figure A2.6 for \( n = 0, 1, \ldots, 10 \). The availability, as given by Equation (A2.43) is obtained as the sum of probabilities. Thus, for \( t = 100 \), we have

\[
A(100) = \sum_{n=0}^{10} P_{10,n}(100) = 0.5612
\]

Similarly

\[
A(500) = \sum_{n=0}^{10} P_{10,n}(500) = 0.6819
\]

Values of \( A(t) \) for various \( t \) are also plotted in Figure A2.6.

### A2.4.3 Average Availability

A sampling measure for the availability of an operational system is the ratio of total up time to total time elapsed. From a practical point of view, it is an important measurable sampling characteristic.

From the definition of availability, we find that the expected value of total up-time by time \( t \) can be expressed as

\[
U(t) = \int_0^t A(x) \, dx.
\]

The ratio of this value to the total time elapsed, \( t \), will give us an average availability up to time \( t \), \( A_{av}(t) \), i.e.

\[
A_{av}(t) = \frac{\int_0^t A(x) \, dx}{t} = \frac{U(t)}{t}
\]

Similarly, the average unavailability can be expressed as

\[
1 - A_{av}(t) = 1 - \frac{\int_0^t (1 - A(x)) \, dx}{t} = \frac{0}{t}.
\]
A2.5 Expected Number of Software, Hardware and Total Failures by Time $t$

A2.5.1 Expected Number of Software Failures

Let $M_s(t)$ be the expected number of software failures detected by time $t$. In order to find the expression for $M_s(t)$, we consider a counting process $\{N_{si}(t), t \geq 0\}$, where $N_{si}(t)$ is the number of software failures detected during the time interval $(0, t]$, when the initial number of errors in the software system is $i$. Let

$$M_{si}(t) = \mathbb{E}[N_{si}(t) | X(0) = i].$$

Then, by conditioning on the first passage time going from state $N$ to $i$,

$$M_s(t) = \sum_{i=0}^{N} M_{si} \cdot G_{N,i}(t)$$

(A2.45)

where $M_{si}(t)$ can be obtained by conditioning on the first down cycle of the process

$$M_{si}(t) = Q_{i,s}(t) + Q_{i,s} Q_{i,s} M_{si}(t)$$

$$+ Q_{i,h} Q_{i,h} M_{si}(t)$$

The Laplace Stieltjes transform of $M_{si}(t)$ is

$$\tilde{M}_{si}(s) = \frac{\lambda_i}{s + \beta + \lambda_i} + a_i \tilde{M}_{si}(s)$$

where $a_i$ is defined in Equation (2.25). Now,

$$\tilde{M}_{si}(s) = \frac{\lambda_i}{s + \beta + \lambda_i} \frac{1}{1 - a_i}$$

or

$$\tilde{M}_{si}(s) = \frac{\lambda_i (s+\nu_i)(s+p_h \gamma)}{(s+x_{1,i})(s+x_{2,i})(s+x_{3,i})}$$

A-32
where $x_{1,i}, x_{2,i}, x_{3,i}$ are given in Equation (2.28).

In a simplified form,

$$\tilde{M}_{si}(s) = \frac{(s + \mu_i)}{p_s u_i} \tilde{G}_{i,i-1}(s) \quad (A2.46)$$

From (A2.45) and (A2.46) the L-S transform for $M_s(t)$ is

$$\tilde{M}_s(s) = \sum_{i=1}^{N} \tilde{M}_{si} \tilde{G}_{N,i}(s)$$

$$= \sum_{i=1}^{N} \frac{(s + \mu_i)}{p_s u_i} \tilde{G}_{N,i-1}(s) \quad (A2.47)$$

Finally, using Lemmas A2.1 to A2.3, we obtain the expression for $M_s(t)$ as

$$M_s(t) = \sum_{i=1}^{N} \sum_{j=1}^{K_i} \frac{U^N_i(-x_j + p_{hi}Y)^{N-i+1}(-x_j + \mu_i)}{p_s u_i} \prod_{l=1}^{K_i} (-x_j + x_l) \quad (1 - e^{-x_j t})$$

where $K_i = 3(N-i+1)$.

A2.5.2 Expected Number of Hardware Failures

Let $M_h(t)$ be the expected number of hardware failures detected by time $t$. Consider a counting process, $\{N_{hi}(t), t \geq 0\}$, where $N_{hi}(t)$ is the number of hardware failures detected during the time interval $(0,t)$, when the initial number of errors in the software system is $i$. Let

$$M_{hi}(t) = E[N_{hi}(t)|X(0) = i]$$

Then, by conditioning on the first passage time going from state $N$ to $i$,

$$M_h(t) = \sum_{i=0}^{N} M_{hi} \ast C_{N,i}(t)$$

where $M_{hi}(t)$ can be obtained by conditioning on the first down cycle of the process.
\[ M_{hi}(t) = Q_i, i_h(t) + Q_i, i_h * Q_H * Q_i, i * M_{hi}(t) + Q_i, i_s * Q_s, i * M_{hi}(t) \]

Now, the L-S transform of \( M_{hi}(t) \) for \( i = 1, 2, \ldots, N \) is

\[ \tilde{M}_{hi}(s) = \frac{\beta}{s + \beta + \lambda_i} + a_i \tilde{M}_{hi}(s) \]

or

\[ \tilde{M}_{hi}(s) = \frac{\beta(s + \mu_i)}{p_s \lambda_i \mu_i} \tilde{G}_{i,i-1}(s), \quad (A2.50) \]

For \( i = 0 \), this L-S transform becomes

\[ \tilde{M}_{h0}(s) = \frac{\beta}{s + \beta} \frac{1}{1 - a_0} \]

or

\[ \tilde{M}_{h0}(s) = \frac{(s + p_h \gamma)}{s(s + \beta + p_h \gamma)} \cdot \]

From (A2.49) the L-S transform for \( M_h(t) \) is

\[ \tilde{M}_h(s) = \sum_{i=0}^{N} \tilde{M}_{hi}(s) \tilde{G}_{N,i}(s) \quad (A2.51) \]

The inverse L-S transform of \( \tilde{M}_{hi}(s) \tilde{G}_{N,i}(s) \) is

\[ G_i(t) = K_i \sum_{j=1}^{N} (-x_j + p_h \gamma)^{N-i+1} (-x_j + \mu_i) \left( \frac{l-e^{-x_j t}}{x_j} \right) \]

where \( K_i = 3(N-i+1) \), and the inverse L-S transform of \( \tilde{M}_{h0}(s) \tilde{G}_{N,0}(s) \) is

A-34
\[
G_0(t) = \frac{\beta u_1^N(p_1^N)^{N+1}}{K-1} + \sum_{j=1}^{K-1} \frac{\beta u_1^N(x_j p_1^N)^{N+1}}{K-1} \prod_{\substack{j=1 \to x_j \neq 0 \to \prod_{l=1 \to x_j \neq 0 \to \prod_{l \neq j}}} (x_j-x_l) \cdot (1-e^{-x_j t})
\]

where \( K = 3N + 2 \).

Finally, the expression for \( M_h(t) \) is

\[
M_h(t) = G_0(t) + \sum_{i=1}^{N} G_i(t).
\]

### A2.5.3 Expected Number of Total Failures

Let \( M(t) \) be the expected number of total failures detected by time \( t \).

Consider \( M_i(t) \) to be the expected number of total failures when there are \( i \) software errors in the system. For any \( i = 0, 1, 2, \ldots, N \),

\[
M_i(t) = M_{si}(t) + M_{hi}(t)
\]

where \( M_{s0}(t) = 0 \),

and \( \hat{M}_i(s) = \hat{M}_{si}(s) + \hat{M}_{hi}(s) \)

Then

\[
M(t) = \sum_{i=0}^{N} M_i(t) G_{N,i}(t), \quad (A2.53)
\]

\[
\hat{M}(s) = \sum_{i=0}^{N} \hat{M}_i(s) \mathcal{G}_{N,i}(s)
\]

or

\[
\hat{M}(s) = \sum_{i=0}^{N} (\hat{M}_{si}(s) + \hat{M}_{hi}(s)) \mathcal{G}_{N,i}(s)
\]

and

\[
M(t) = M_s(t) + M_h(t) \quad (A2.54)
\]
<table>
<thead>
<tr>
<th>TIME</th>
<th>SOFTWARE FAILURES</th>
<th>HARDWARE FAILURES</th>
<th>TOTAL FAILURES</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>20.00</td>
<td>2.48</td>
<td>0.14</td>
<td>2.62</td>
</tr>
<tr>
<td>40.00</td>
<td>4.19</td>
<td>0.26</td>
<td>4.45</td>
</tr>
<tr>
<td>60.00</td>
<td>5.47</td>
<td>0.38</td>
<td>5.85</td>
</tr>
<tr>
<td>80.00</td>
<td>6.47</td>
<td>0.49</td>
<td>6.96</td>
</tr>
<tr>
<td>100.00</td>
<td>7.27</td>
<td>0.61</td>
<td>7.88</td>
</tr>
<tr>
<td>120.00</td>
<td>7.92</td>
<td>0.72</td>
<td>8.64</td>
</tr>
<tr>
<td>160.00</td>
<td>8.91</td>
<td>0.94</td>
<td>9.85</td>
</tr>
<tr>
<td>200.00</td>
<td>9.59</td>
<td>1.17</td>
<td>10.77</td>
</tr>
<tr>
<td>240.00</td>
<td>10.06</td>
<td>1.41</td>
<td>11.48</td>
</tr>
<tr>
<td>280.00</td>
<td>10.39</td>
<td>1.66</td>
<td>12.05</td>
</tr>
<tr>
<td>320.00</td>
<td>10.62</td>
<td>1.92</td>
<td>12.53</td>
</tr>
<tr>
<td>360.00</td>
<td>10.77</td>
<td>2.18</td>
<td>12.95</td>
</tr>
<tr>
<td>400.00</td>
<td>10.88</td>
<td>2.44</td>
<td>13.32</td>
</tr>
<tr>
<td>440.00</td>
<td>10.95</td>
<td>2.71</td>
<td>13.66</td>
</tr>
<tr>
<td>480.00</td>
<td>11.00</td>
<td>2.98</td>
<td>13.99</td>
</tr>
<tr>
<td>520.00</td>
<td>11.04</td>
<td>3.25</td>
<td>14.29</td>
</tr>
<tr>
<td>560.00</td>
<td>11.06</td>
<td>3.53</td>
<td>14.59</td>
</tr>
<tr>
<td>600.00</td>
<td>11.08</td>
<td>3.80</td>
<td>14.88</td>
</tr>
<tr>
<td>640.00</td>
<td>11.09</td>
<td>4.08</td>
<td>15.17</td>
</tr>
<tr>
<td>680.00</td>
<td>11.10</td>
<td>4.35</td>
<td>15.45</td>
</tr>
<tr>
<td>720.00</td>
<td>11.10</td>
<td>4.63</td>
<td>15.73</td>
</tr>
<tr>
<td>760.00</td>
<td>11.10</td>
<td>4.91</td>
<td>16.01</td>
</tr>
<tr>
<td>800.00</td>
<td>11.11</td>
<td>5.18</td>
<td>16.29</td>
</tr>
<tr>
<td>840.00</td>
<td>11.11</td>
<td>5.46</td>
<td>16.57</td>
</tr>
<tr>
<td>880.00</td>
<td>11.11</td>
<td>5.74</td>
<td>16.85</td>
</tr>
<tr>
<td>920.00</td>
<td>11.11</td>
<td>6.01</td>
<td>17.12</td>
</tr>
<tr>
<td>960.00</td>
<td>11.11</td>
<td>6.29</td>
<td>17.40</td>
</tr>
<tr>
<td>1000.00</td>
<td>11.11</td>
<td>6.57</td>
<td>17.68</td>
</tr>
</tbody>
</table>
i.e. the expected number of total failures detected by time \( t \) is equal to the sum of the expected number of software and hardware failures.

A2.5.4 **Illustrative Example**

Consider a system with an initial number of software errors, \( N = 10 \); probabilities of perfect software and hardware maintenance \( p_s = .9 \) and \( p_h = .9 \), respectively; software failure rate \( \lambda_i = i\lambda \), and \( \lambda = .02 \); software repair rate \( \mu_i = i\mu \), and \( \mu = .05 \); and hardware failure and repair rates \( \beta = .01 \) and \( \gamma = .025 \), respectively.

For this system, the expected number of software, hardware, and total system failures are computed from Equations (A2.45), (A2.49), and (A2.54), respectively. Selected values of these quantities are given in Table A2.3 and plotted in Figure A2.7. The information in Table A2.3 shows us that the number of software failures detected is increasing rapidly at the early times and then slows down as the number of remaining software errors and software failure rate decrease.

On the other hand, the number of hardware failures detected is increasing with the slow-down of the number of software failures detected.
Figure A2.7. Expected Cumulative Number of Failures.
A3. OPERATIONAL COST MODELS

In Section 2 we proposed a model for the operational phase of the hardware-software system and developed expressions for several performance measures. In many applications, these individual measures are of less interest than an overall measure, such as the expected total cost. With this objective, in this section we develop cost models for the hardware, software, and hardware-software systems. The principal cost components considered are the cost of a failure, the cost of the maintenance activity performed to bring the system back to an operational state, and the cost of system downtime. The primary measures that affect the total cost are the number of failures and the system availability.

The relative importance of these measures in a given situation can be expressed via the numerical values for the cost factors.

Models for hardware and software systems are developed in Sections A3.1 and A3.2, respectively. The total hardware-software system is discussed in Section A3.3. Several numerical examples are used to illustrate the results.
A3.1 Operational Cost Model: Hardware System

In this Section we develop a cost model for the hardware system. The system is in an up state at time \( t = 0 \). After a random time \( U \), whose distribution is exponential with parameter \( \beta \) (see Equation A2.5), a failure occurs and the system goes into a down state. A repair or maintenance activity is undertaken and after a random time \( V \), whose distribution is exponential with parameter \( \gamma \) (see Equation A2.6), the system is brought back into an up state. The cause of the failure would have been removed with probability \( p_h \) (\( 0 \leq p_h \leq 1 \)). The sequence of up and down states forms a renewal process. For purposes of this Section, it is assumed that the software system has no effect on the operation of the hardware system.

The following costs are incurred due to the failure and maintenance activities:

(i) A fixed cost \( c_{h_1} \) per failure
(ii) A variable cost \( c_{h_2} \) per repair per unit time
(iii) A variable cost due to the unavailability of the system, \( c_{h_3} \) per unit time.

Consider the time interval \((0,t)\).

Let

\[ C_h(t) = \text{expected total cost incurred by } t, \]
\[ M_h(t) = \text{expected number of hardware failures by } t, \]
Let $A_h(t)$ denote system availability at $t$.

Then, the expected total cost by time $t$ is given by

$$C_h(t) = c_h M_h(t) + c_h y t + c_h \int_0^t (1-A_h(x)) dx \quad (A3.1)$$

where

$$\int_0^t (1-A_h(x)) dx$$

is the expected total down time during $(0,t)$. Now we develop expressions for $M_h(t)$ and $A_h(t)$ and obtain a closed form equation for $C_h(t)$. Consider one up and down cycle, i.e., one renewal. If the maintenance activity is perfect, the length of this cycle will be $U + V$. If, however, the maintenance activity is imperfect, the repair will go another $V$ units of time so that the length of the cycle will be $U + V + V$. If the maintenance is imperfect for the second time, the length of the cycle will be $U + V + V + V$, and so on.

Therefore, the probability density function, $g$, of the renewal time is given by

$$g = p_h f_u * f_v + p_h q_h f_u * f_v * f_v + p_h^2 f_u * f_v * f_v * f_v + \ldots, \quad (A3.2)$$

where * stands for convolution,

$f_u$ is the pdf of $U$,
and \( f_V \) is the pdf of \( V \).

The Laplace transform of \( g, g^* \), is

\[
g^*(s) = p_h f_U^*(s) f_V^*(s) [1 + q_h f_V^*(s) + (q_h f_V^*(s))^2 + \ldots ]
\]

or

\[
g^*(s) = p_h f_U^*(s) f_V^*(s) \left( \frac{1}{1 - q_h f_V^*(s)} \right)
\]

or

\[
g^*(s) = p_h \frac{\beta}{s + \beta} \frac{\gamma}{s + \gamma} \frac{s + \gamma}{s + p_h \gamma}
\]

or

\[
g^*(s) = \frac{\beta p_h \gamma}{(s + \beta)(s + p_h \gamma)} \quad (A3.3)
\]

Now, the renewal equations for the expected number of hardware failures can be written as

\[
M_h(t) = F_U(t) + \int_0^t M_h(t - x) g(x) \, dx \quad (A3.4)
\]

where \( F_U(t) \) is the cdf of \( U \).

The Laplace transform of \( M_h(t) \) is

\[
M_h^*(s) = \frac{f_U^*(s)}{s} + M_h^*(s) g^*(s)
\]

\[
= \frac{\beta}{s(s + \beta)} + M_h^*(s) \frac{\beta}{s + \beta} \frac{p_h \gamma}{s + p_h \gamma}
\]

or

\[
M_h^*(s) = \frac{\beta(s + p_h \gamma)}{s^2(s + \beta + p_h \gamma)}
\]

By taking the inverse Laplace transform we get
The renewal equation for $A_h(t)$ can be written as

$$A_h(t) = 1 - F_U(t) + \int_0^t A_h(t-x)g(x)dx,$$

and its Laplace transform as

$$A_h^*(s) = \frac{1 - f_u^*(s)}{s[1 - g^*(s)]} = \frac{s + p_h\gamma}{s(s + \beta + p_h\gamma)}.$$

Therefore, the availability of the system at time $t$ is

$$A_h(t) = \mathcal{L}^{-1} A_h^*(s) = e^{-(\beta+\gamma)t} + p_h\gamma \left[ \frac{1 - e^{-(\beta+p_h\gamma)t}}{\beta + p_h\gamma} \right],$$

or

$$A_h(t) = \frac{p_h\gamma + \beta e^{-(\beta+p_h\gamma)t}}{\beta + p_h\gamma}. \quad (A3.6)$$

Now, the expected total down time during $(0,t)$ is
\[ \int_0^t (1 - A_h(x)) \, dx \text{ which, on substituting for } A_h(x) \text{ from Equation (A3.6), gives} \]

\[ \int_0^t (1 - A_h(x)) \, dx = \beta \left[ \frac{-(\beta + p_h \gamma) t - 1 + e}{(\beta + p_h \gamma)^2} \right] \quad (A3.7) \]

On substituting the expressions for \( M_h(t) \) and \( \int_0^t (1 - A_h(x)) \, dx \) from Equations (A3.5) and (A3.7), respectively, in Equation (3.1), we get, after some simplification,

\[ C_h(t) = \frac{c_{h1} \beta}{(\beta + p_h \gamma)^2} \left[ p_h \gamma (\beta + p_h \gamma) t + \beta (1 - e^{-(\beta + p_h \gamma) t}) \right] + c_{h2} \gamma t \]

\[ + \frac{c_{h3} \beta}{(\beta + p_h \gamma)^2} \left[ (\beta + p_h \gamma) t - 1 + e^{-(\beta + p_h \gamma) t} \right] \quad (A3.8) \]

The above equation gives the expected cost incurred by time \( t \) in terms of the hardware system parameters \( \beta, \gamma, \) and \( p_h \), and the cost factors \( c_{h1}', c_{h2}', \) and \( c_{h3}' \).
Illustrative Examples

We numerically study the behavior of $M_h(t)$, $A_{\text{nav}}(t)$ and $C_h(t)/t$ as a function of the cost factors $c_{h_1}$, $c_{h_2}$, $c_{h_3}$ and of the failure and repair rates $\beta$ and $\gamma$, respectively.

Consider a system with $\beta = .01$, $p_h = 0.9$, and $\gamma = .01$, .02, .05, .10, .20, .30, .50, .75, 1.0, 2.0, 3.0, and 4.0. The average availability ($A_{\text{nav}}(t)$) and the expected number of failures by time $t$ are shown in Table A3.1 for $t = 100$, 250, 500, 1000, and 2000. We notice that for a fixed repair rate the average availability decreases with time, the rate of decrease being higher for low values of $\gamma$. The expected number of failures in a given time interval increases with $\gamma$. This is so because at low values of $\gamma$, the system is down for longer periods of time, causing a reduction in the up time of the system.

The expected total cost per unit time ($C_h(t)/t$) is now calculated from Equation (A3.8) for given cost factors. Such values for four sets of cost factors are given in Table A3.2. For a given $t$, the cost first decreases and then increases as a function of $\gamma$. In other words, $C_h(t)/t$ seems to be a convex function with respect to $\gamma$. 

A-45
TABLE A3.1

AVERAGE AVAILABILITY AND EXPECTED NUMBER OF FAILURES

HARDWARE SYSTEM
FAILURE RATE: 0.010

AVERAGE AVAILABILITY

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>0.709261</td>
<td>0.583529</td>
<td>0.529082</td>
<td>0.501385</td>
<td>0.487535</td>
</tr>
<tr>
<td>0.02</td>
<td>0.762652</td>
<td>0.693831</td>
<td>0.668367</td>
<td>0.655612</td>
<td>0.649235</td>
</tr>
<tr>
<td>0.05</td>
<td>0.851105</td>
<td>0.831405</td>
<td>0.824793</td>
<td>0.821488</td>
<td>0.819835</td>
</tr>
<tr>
<td>0.10</td>
<td>0.910000</td>
<td>0.904000</td>
<td>0.902000</td>
<td>0.901000</td>
<td>0.900500</td>
</tr>
<tr>
<td>0.20</td>
<td>0.950139</td>
<td>0.948476</td>
<td>0.947222</td>
<td>0.947645</td>
<td>0.947507</td>
</tr>
<tr>
<td>0.30</td>
<td>0.965561</td>
<td>0.964796</td>
<td>0.964541</td>
<td>0.964413</td>
<td>0.964349</td>
</tr>
<tr>
<td>0.50</td>
<td>0.978733</td>
<td>0.978450</td>
<td>0.978355</td>
<td>0.978308</td>
<td>0.978284</td>
</tr>
<tr>
<td>0.75</td>
<td>0.985615</td>
<td>0.985487</td>
<td>0.985444</td>
<td>0.985423</td>
<td>0.985412</td>
</tr>
<tr>
<td>1.00</td>
<td>0.989132</td>
<td>0.989059</td>
<td>0.989035</td>
<td>0.989023</td>
<td>0.989017</td>
</tr>
<tr>
<td>1.50</td>
<td>0.992701</td>
<td>0.992669</td>
<td>0.992658</td>
<td>0.992652</td>
<td>0.992650</td>
</tr>
<tr>
<td>2.00</td>
<td>0.994506</td>
<td>0.994487</td>
<td>0.994481</td>
<td>0.994478</td>
<td>0.994477</td>
</tr>
<tr>
<td>3.00</td>
<td>0.996324</td>
<td>0.996315</td>
<td>0.996313</td>
<td>0.996311</td>
<td>0.996311</td>
</tr>
<tr>
<td>4.00</td>
<td>0.997238</td>
<td>0.997233</td>
<td>0.997231</td>
<td>0.997231</td>
<td>0.997230</td>
</tr>
</tbody>
</table>

EXPECTED NUMBER OF HARDWARE FAILURES

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>0.01</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.30</th>
<th>0.50</th>
<th>0.75</th>
<th>1.00</th>
<th>1.50</th>
<th>2.00</th>
<th>3.00</th>
<th>4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.7093</td>
<td>1.4588</td>
<td>2.6454</td>
<td>5.0139</td>
<td>9.7507</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.7627</td>
<td>1.7346</td>
<td>3.3418</td>
<td>6.5561</td>
<td>12.9847</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.8511</td>
<td>2.0785</td>
<td>4.1240</td>
<td>8.2149</td>
<td>16.3967</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9100</td>
<td>2.2600</td>
<td>4.5100</td>
<td>9.0100</td>
<td>18.0100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9501</td>
<td>2.3712</td>
<td>4.7396</td>
<td>9.4765</td>
<td>18.9501</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9656</td>
<td>2.4120</td>
<td>4.8227</td>
<td>9.6441</td>
<td>19.2870</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9787</td>
<td>2.4461</td>
<td>4.8918</td>
<td>9.7831</td>
<td>19.5657</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9856</td>
<td>2.4637</td>
<td>4.9272</td>
<td>9.8542</td>
<td>19.7082</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9891</td>
<td>2.4726</td>
<td>4.9452</td>
<td>9.8902</td>
<td>19.7803</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9927</td>
<td>2.4817</td>
<td>4.9633</td>
<td>9.9265</td>
<td>19.8530</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9945</td>
<td>2.4862</td>
<td>4.9724</td>
<td>9.9448</td>
<td>19.8895</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9963</td>
<td>2.4908</td>
<td>4.9816</td>
<td>9.9631</td>
<td>19.9262</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.9972</td>
<td>2.4931</td>
<td>4.9862</td>
<td>9.9723</td>
<td>19.9446</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPAIR RATE</td>
<td>TIME</td>
<td>100.0</td>
<td>250.0</td>
<td>500.0</td>
<td>1000.0</td>
<td>2000.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td></td>
<td>3.0783</td>
<td>4.3231</td>
<td>4.8621</td>
<td>5.1363</td>
<td>5.2734</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>2.6497</td>
<td>3.3311</td>
<td>3.5832</td>
<td>3.7094</td>
<td>3.7726</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>2.0741</td>
<td>2.2691</td>
<td>2.3345</td>
<td>2.3673</td>
<td>2.3836</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td></td>
<td>1.9910</td>
<td>2.0504</td>
<td>2.0702</td>
<td>2.0801</td>
<td>2.0850</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td></td>
<td>2.5936</td>
<td>2.6101</td>
<td>2.6156</td>
<td>2.6183</td>
<td>2.6197</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td></td>
<td>3.4409</td>
<td>3.4485</td>
<td>3.4510</td>
<td>3.4523</td>
<td>3.4529</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td></td>
<td>5.3105</td>
<td>5.3133</td>
<td>5.3143</td>
<td>5.3147</td>
<td>5.3150</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>7.7424</td>
<td>7.7437</td>
<td>7.7441</td>
<td>7.7443</td>
<td>7.7444</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>10.2076</td>
<td>10.2083</td>
<td>10.2086</td>
<td>10.2087</td>
<td>10.2087</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.50</td>
<td></td>
<td>15.1723</td>
<td>15.1726</td>
<td>15.1727</td>
<td>15.1727</td>
<td>15.1728</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.00</td>
<td></td>
<td>20.1544</td>
<td>20.1546</td>
<td>20.1546</td>
<td>20.1547</td>
<td>20.1547</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td>30.1364</td>
<td>30.1365</td>
<td>30.1365</td>
<td>30.1365</td>
<td>30.1365</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td></td>
<td>40.1273</td>
<td>40.1274</td>
<td>40.1274</td>
<td>40.1274</td>
<td>40.1274</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>TIME</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td></td>
<td>3.7167</td>
<td>4.8482</td>
<td>5.3383</td>
<td>5.5875</td>
<td>5.7122</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>3.3361</td>
<td>3.9555</td>
<td>4.1847</td>
<td>4.2995</td>
<td>4.3569</td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>2.8401</td>
<td>3.0174</td>
<td>3.0769</td>
<td>3.1066</td>
<td>3.1215</td>
</tr>
<tr>
<td>0.10</td>
<td></td>
<td>2.8100</td>
<td>2.8640</td>
<td>2.8820</td>
<td>2.8910</td>
<td>2.8955</td>
</tr>
<tr>
<td>0.20</td>
<td></td>
<td>3.4488</td>
<td>3.4637</td>
<td>3.4687</td>
<td>3.4712</td>
<td>3.4724</td>
</tr>
<tr>
<td>0.30</td>
<td></td>
<td>4.3099</td>
<td>4.3168</td>
<td>4.3191</td>
<td>4.3203</td>
<td>4.3209</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>8.6295</td>
<td>8.6306</td>
<td>8.6310</td>
<td>8.6312</td>
<td>8.6313</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>11.0978</td>
<td>11.0985</td>
<td>11.0987</td>
<td>11.0988</td>
<td>11.0988</td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td>31.0331</td>
<td>31.0332</td>
<td>31.0332</td>
<td>31.0332</td>
<td>31.0332</td>
</tr>
<tr>
<td>4.00</td>
<td></td>
<td>41.0249</td>
<td>41.0249</td>
<td>41.0249</td>
<td>41.0249</td>
<td>41.0249</td>
</tr>
</tbody>
</table>

A-47
<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>TIME</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td></td>
<td>29.2448</td>
<td>41.8055</td>
<td>47.2447</td>
<td>50.0116</td>
<td>51.3953</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>24.0111</td>
<td>30.8863</td>
<td>33.4301</td>
<td>34.7043</td>
<td>35.3415</td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>15.4747</td>
<td>17.4426</td>
<td>18.1031</td>
<td>18.4334</td>
<td>18.5985</td>
</tr>
<tr>
<td>0.10</td>
<td></td>
<td>10.0910</td>
<td>10.6904</td>
<td>10.8902</td>
<td>10.9901</td>
<td>11.0401</td>
</tr>
<tr>
<td>0.20</td>
<td></td>
<td>7.0812</td>
<td>7.2472</td>
<td>7.3025</td>
<td>7.3302</td>
<td>7.3441</td>
</tr>
<tr>
<td>0.30</td>
<td></td>
<td>6.5404</td>
<td>6.6169</td>
<td>6.6424</td>
<td>6.6551</td>
<td>6.6615</td>
</tr>
<tr>
<td>0.50</td>
<td></td>
<td>7.2245</td>
<td>7.2529</td>
<td>7.2623</td>
<td>7.2670</td>
<td>7.2694</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>9.0371</td>
<td>9.0499</td>
<td>9.0541</td>
<td>9.0563</td>
<td>9.0573</td>
</tr>
<tr>
<td>1.50</td>
<td></td>
<td>15.8292</td>
<td>15.8324</td>
<td>15.8335</td>
<td>15.8340</td>
<td>15.8343</td>
</tr>
<tr>
<td>2.00</td>
<td></td>
<td>20.6489</td>
<td>20.6507</td>
<td>20.6513</td>
<td>20.6516</td>
<td>20.6518</td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td>30.4673</td>
<td>30.4681</td>
<td>30.4684</td>
<td>30.4685</td>
<td>30.4686</td>
</tr>
<tr>
<td>4.00</td>
<td></td>
<td>40.3760</td>
<td>40.3764</td>
<td>40.3766</td>
<td>40.3767</td>
<td>40.3767</td>
</tr>
</tbody>
</table>

**CH1=10, CH2=100, AND CH3=100**

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>TIME</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td></td>
<td>3.9783</td>
<td>5.2231</td>
<td>5.7621</td>
<td>6.0363</td>
<td>6.1734</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>4.4497</td>
<td>5.1311</td>
<td>5.3832</td>
<td>5.5094</td>
<td>5.5726</td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>6.5741</td>
<td>6.7691</td>
<td>6.8345</td>
<td>6.8673</td>
<td>6.8836</td>
</tr>
<tr>
<td>0.10</td>
<td></td>
<td>10.9910</td>
<td>11.0504</td>
<td>11.0702</td>
<td>11.0801</td>
<td>11.0851</td>
</tr>
<tr>
<td>0.20</td>
<td></td>
<td>20.5936</td>
<td>20.6101</td>
<td>20.6156</td>
<td>20.6183</td>
<td>20.6197</td>
</tr>
<tr>
<td>0.30</td>
<td></td>
<td>30.4409</td>
<td>30.4485</td>
<td>30.4510</td>
<td>30.4523</td>
<td>30.4529</td>
</tr>
<tr>
<td>0.50</td>
<td></td>
<td>50.3105</td>
<td>50.3133</td>
<td>50.3143</td>
<td>50.3147</td>
<td>50.3150</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>75.2424</td>
<td>75.2437</td>
<td>75.2441</td>
<td>75.2443</td>
<td>75.2444</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>100.2076</td>
<td>100.2083</td>
<td>100.2086</td>
<td>100.2087</td>
<td>100.2087</td>
</tr>
<tr>
<td>1.50</td>
<td></td>
<td>150.1723</td>
<td>150.1726</td>
<td>150.1727</td>
<td>150.1727</td>
<td>150.1728</td>
</tr>
<tr>
<td>2.00</td>
<td></td>
<td>200.1544</td>
<td>200.1546</td>
<td>200.1546</td>
<td>200.1547</td>
<td>200.1547</td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td>300.1364</td>
<td>300.1365</td>
<td>300.1365</td>
<td>300.1365</td>
<td>300.1365</td>
</tr>
<tr>
<td>4.00</td>
<td></td>
<td>400.1273</td>
<td>400.1274</td>
<td>400.1274</td>
<td>400.1274</td>
<td>400.1274</td>
</tr>
</tbody>
</table>

**CH1=10, CH2=10, AND CH3=10**
Plots of $A_{hav}(t)$ for $\beta = .01, .05, \text{ and } 0.10$ versus $\gamma$ are shown in Figure A3.1. As expected, the average availability improves with $\gamma$ as well as with an improvement in the failure rate, i.e., as $\beta$ goes from 0.10 to 0.01.

Costs per unit times for various $t$ are shown in Figure A3.2 for $\beta = .01$, $c_{h_1} = 10$, $c_{h_2} = 10$, $c_{h_3} = 100$ as a function of $\gamma$ and clearly show the convexity of the cost function. A similar pattern is seen in Figure A3.3 which gives the plots of $C_h(t)/t$ for the four sets of cost factors at time $t = 500$ and $\beta = .01$. 
Figure A3.1. Average Availability vs. Repair Rate for Different Failure Rates. (t = 500)
Figure A3.2. Expected Total Cost/Unit Time vs. Repair Rate for Different Cost Factors ($\beta = .01$, $t = 500$).
Figure A3.3. Expected Total Cost/Unit Time vs. Repair Rate for Different Times ($\beta = .01$, $c_{h_1} = 10$, $c_{r_2} = 10$, $c_{h_3} = 100$).
A3.2 Operational Cost Model: Software System

Consider a system consisting of software only. At time zero it is operational with \( N \) errors in the system. A failure occurs at a random time, \( T_N \), whose distribution is given by Equation A2.2 with parameter \( \lambda_N \). A repair is undertaken and with probability \( p_s \) the error causing the failure is removed in a time \( W_N \) whose distribution is exponential with parameter \( \mu_N \) (see Equation A2.4). The next cycle starts with \((N-1)\) errors in the system and the failure distribution is now exponential with a parameter \((N-1)\lambda\). If the error is not removed, which happens with probability \( q_s = 1 - p_s \), the distribution of time to next failure is again exponential with parameter \( \lambda_N \). A similar behavior is observed throughout the entire life cycle of the software system with \( i(0 \leq i \leq N) \) remaining errors. Note that the model is similar to the Imperfect Maintenance Model (IMM) of Okumoto and Goel (1978).

A diagrammatic representation of the behavior of the software system is shown in Figure A3.4.

As discussed in Section A3.1, for a hardware system, the cost elements associated with the failure-repair cycles of the software system are

\[ c_{S1} = \text{cost of a software failure}, \]
Software System: Diagrammatic Representation of Transitions between States of $X(t)$.
\[ c_{s_2} = \text{cost incurred per repair per unit time}, \]
and \[ c_{s_3} = \text{cost of system down time per unit time}. \]

Then, the expected total operational cost by time \( t \) is given by

\[
C_s(t) = c_{s_1} M_s(t) + c_{s_2} \mu t + c_{s_3} \int_0^t (1 - A_s(x)) \, dx \quad (A3.9)
\]

where

\[ M_s(t) = \text{expected number of software failures by time } t, \]
\[ A_s(t) = \text{availability of the software system at time } t. \]

To get the expression for \( M_s(t) \) and \( A_s(t) \), we first give the Laplace-Stieltjes transforms of the appropriate quantities as follows.

Let \( G_{N,i}(t) \) be the distribution function of the first passage time from state \( N \) to state \( i \). By considering the renewal equation associated with this, the Laplace-Stieltjes transform of \( G_{N,i}(t) \) is obtained as

\[
\tilde{G}_{N,i}(s) = \prod_{j=i+1}^{N} \frac{p_s \lambda_j \mu_j}{s^2 + s(\lambda_j + \mu_j) + p\lambda_j \mu_j} \quad (A3.10)
\]

Similarly, the L-S transforms of \( M_s(t) \) and \( A_s(t) \) are given by

\[
\tilde{M}_s(s) = \sum_{i=1}^{N} \frac{\lambda_i (s + \mu_i)}{s^2 + s(\lambda_i + \mu_i) + p_s \lambda_i \mu_i} \tilde{G}_{N,i}(s) \quad (A3.11)
\]

and

\[ A-55 \]
\[ \tilde{A}_s(s) = \sum_{i=0}^{N} \left( 1 - \frac{\lambda_i(s + p_i\mu_i)}{s^2 + s(\lambda_i + \mu_i) + p_i\lambda_i\mu_i} \right) \tilde{G}_{N,i}(s) \]  
(A3.12)

where \( \tilde{G}_{N,N}(s) = 1 \).

Unlike the hardware system discussed in the previous Section, the results for \( M_s(t) \) and \( A_s(t) \) cannot be obtained in a closed form, but can be derived from Equations (A3.11) and (A3.12) by using Lemmas A2.1, A2.2, and A2.3 as follows.

First we obtain the inverse Laplace-Stieltjes transform for Equation (A3.10). We write

\[ \tilde{G}_{N,i}(s) = \prod_{j=i+1}^{N} \frac{p_s\lambda_j\mu_j}{(s + x_{1,j})(s + x_{2,j})} \]

Let \( x_{1,i+1} = x_{1}, x_{2,i+1} = x_{2}, x_{1,i+2} = x_{3}, x_{2,i+2} = x_{4}, \ldots, x_{2,N} = x_{K_i} \) where \( K_i = (N-i) \times 2 \), and let

\[ U_i = \prod_{j=i+1}^{N} p_s\lambda_j\mu_j, \]

as given in Equation (A2.33)

By Lemmas A2.1-A2.3

\[ \begin{align*}
G_{N,i}(t) &= \sum_{j=1}^{K_i} \left( \prod_{k=1}^{N} p_s\lambda_k\mu_k \right) \frac{\frac{-x_j t}{\prod_{l=1}^{K_i} (-x_j + x_{l})}}{-x_j} \prod_{l=1}^{K_i} (-x_j + x_{l}) \\
&= \sum_{j=1}^{K_i} \frac{\frac{-x_j t}{\prod_{l=1}^{K_i} (-x_j + x_{l})}}{U_i} \prod_{l=1}^{K_i} (-x_j + x_{l})
\end{align*} \]

A-56
Similarly, we get

\[ \tilde{M}_s(s) = \sum_{i=1}^{N} \frac{\lambda_i(s + u_i)}{(s + x_{1,i})(s + x_{2,i})} \prod_{j=i+1}^{N} \frac{(p_s \lambda_j u_j)}{(s + x_{1,j})(s + x_{2,j})} \]

\[ = \sum_{i=1}^{N} \left\{ \frac{\lambda_i(s + u_i)}{\prod_{k=i}^{N} (s + x_{1,k})(s + x_{2,k})} \right\} \]

and

\[ M_s(t) = \sum_{i=1}^{N} \left\{ \sum_{j=1}^{K_i} \frac{\lambda_i \prod_{k=i+1}^{N} p_s \lambda_k u_k}{\prod_{k=i+1}^{N} (-x_j + x_k)} \right\} (e^{-x_j t} - 1) \]

\[ + \frac{\lambda_i u_i \prod_{k=i+1}^{N} p_s \lambda_k u_k}{\prod_{k=i+1}^{N} (-x_j + x_k)} \sum_{j=1}^{K_i} \frac{(-x_j + x_k)}{e^{-x_j t}} \]

or

\[ M_s(t) = \sum_{i=1}^{N} \lambda_i \prod_{j=1}^{K_i} \frac{\sum_{k=i+1}^{N} \lambda_k u_k (e^{-x_j t} - 1)}{(-x_j + x_k)} \]

\[ - x_j \sum_{k=1}^{K_i} \frac{(-x_j + x_k)}{e^{-x_j t}} \]

(A3.13)

For the availability, taking the inverse L-S transform of Equation (A3.12), we have
\[ A_s(t) = \sum_{i=0}^{N} \{ G_{N,i}(t) - G_{N,i-1}(t) \} - \lambda_i \sum_{i=1}^{N} \frac{K_i}{\prod_{j=1}^{i} (-x_j + x_i')} \frac{(e^{-x_i't} - 1)}{j \neq \ell} \]  

(A3.14)

For given \( c_{s1}, c_{s2}, \) and \( c_{s3} \), the expected total cost can be obtained from Equation (A3.9) by substituting for \( M_s(t) \) and \( A_s(t) \) from Equations (A3.13) and (A3.14), respectively.

**Illustrative Examples**

Now we numerically study the behavior of \( M_s(t) \), \( A_{sav}(t) \) and \( C_s(t)/t \) as a function of the software repair rate \( \mu \), failure rate \( \lambda \), and the cost factors \( c_{s1}, c_{s2}, \) and \( c_{s3} \).

Let us consider a system with \( N = 10, \mu = 0.05, \) and \( p_s = 0.9. \) The values of \( A_{sav}(t) \) and \( M_s(t) \) computed from the formulae derived earlier in this section are given in Table 3.3 for various values of \( \mu \) and \( t \). From the table we note that the average availability improves with \( t \) as well as with \( \mu \). The improvement with \( t \) is due to the fact that, as more software errors are removed, the system fails less often. The improvement with repair rate is due to shorter down time.

The expected number of failures increases with \( t \) and
with \( \nu \). As the system is used for longer time, more errors surface resulting in more failures. Also, as \( \nu \) improves, the system is up for longer periods of time resulting in more software failures. Note that the asymptotic value of \( M_s(t) \) is simply the ratio \( N/p_s = 10/0.9 = 11.1111 \). Plots of \( A_{sav}(t) \) for \( \lambda = .01, .05, \) and .10 versus \( \nu \) are shown in Figure 3.5. As one would expect, availability improves as \( \nu \) goes up and also as \( \lambda \) goes from .10 to .05 to .01.

The expected total cost per unit time \( C_s(t)/t \), is given in Table A3.4 for \( \lambda = 0.05, N = 10, p_s = 0.9, \nu \) varying from 0.01 to 4.00, \( t \) from 100 to 2000, and the cost factors varying as follows:

\[
\begin{array}{ccc}
C_s^1 & C_s^2 & C_s^3 \\
10 & 10 & 10 \\
100 & 10 & 10 \\
10 & 10 & 100 \\
10 & 100 & 10 \\
\end{array}
\]

Two additional sets of plots of the cost values versus \( \nu \), taken from the above tables, are shown in Figures A3.6 and A3.7.
TABLE A3.3

AVERAGE AVAILABILITY AND EXPECTED NUMBER OF FAILURES
SOFTWARE SYSTEM
FAILURE RATE: 0.050

AVERAGE AVAILABILITY

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>TIME</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td></td>
<td>0.196539</td>
<td>0.215310</td>
<td>0.392714</td>
<td>0.674846</td>
<td>0.837280</td>
</tr>
<tr>
<td>0.02</td>
<td></td>
<td>0.322907</td>
<td>0.424598</td>
<td>0.675202</td>
<td>0.837280</td>
<td>0.918640</td>
</tr>
<tr>
<td>0.05</td>
<td></td>
<td>0.548578</td>
<td>0.740689</td>
<td>0.869824</td>
<td>0.934912</td>
<td>0.967456</td>
</tr>
<tr>
<td>0.10</td>
<td></td>
<td>0.722236</td>
<td>0.869873</td>
<td>0.934912</td>
<td>0.967456</td>
<td>0.983728</td>
</tr>
<tr>
<td>0.20</td>
<td></td>
<td>0.849874</td>
<td>0.934920</td>
<td>0.967456</td>
<td>0.983728</td>
<td>0.991864</td>
</tr>
<tr>
<td>0.30</td>
<td></td>
<td>0.898108</td>
<td>0.956612</td>
<td>0.978304</td>
<td>0.989152</td>
<td>0.994576</td>
</tr>
<tr>
<td>0.50</td>
<td></td>
<td>0.938155</td>
<td>0.973966</td>
<td>0.986982</td>
<td>0.993491</td>
<td>0.996746</td>
</tr>
<tr>
<td>0.75</td>
<td></td>
<td>0.958565</td>
<td>0.982644</td>
<td>0.991322</td>
<td>0.995661</td>
<td>0.997830</td>
</tr>
<tr>
<td>1.00</td>
<td></td>
<td>0.968853</td>
<td>0.986983</td>
<td>0.993491</td>
<td>0.996746</td>
<td>0.998373</td>
</tr>
<tr>
<td>1.50</td>
<td></td>
<td>0.979190</td>
<td>0.991322</td>
<td>0.995661</td>
<td>0.997830</td>
<td>0.998915</td>
</tr>
<tr>
<td>2.00</td>
<td></td>
<td>0.984376</td>
<td>0.993491</td>
<td>0.996746</td>
<td>0.998373</td>
<td>0.999186</td>
</tr>
<tr>
<td>3.00</td>
<td></td>
<td>0.989574</td>
<td>0.995661</td>
<td>0.997830</td>
<td>0.998915</td>
<td>0.999458</td>
</tr>
<tr>
<td>4.00</td>
<td></td>
<td>0.992176</td>
<td>0.996746</td>
<td>0.998373</td>
<td>0.999186</td>
<td>0.999593</td>
</tr>
</tbody>
</table>

EXPECTED NUMBER OF SOFTWARE FAILURES

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>EXP. FAILURES</th>
<th>0.01</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.30</th>
<th>0.50</th>
<th>0.75</th>
<th>1.00</th>
<th>1.50</th>
<th>2.00</th>
<th>3.00</th>
<th>4.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>6.6245</td>
<td>10.1158</td>
<td>11.0700</td>
<td>11.1109</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>8.6796</td>
<td>11.0015</td>
<td>11.1106</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>10.3617</td>
<td>11.1095</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>10.8006</td>
<td>11.1109</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>10.9340</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>10.9650</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>10.9858</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>10.9949</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>10.9992</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.50</td>
<td>11.0034</td>
<td>11.1110</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.00</td>
<td>11.0054</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>11.0073</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td>11.0083</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>REPAIR RATE</td>
<td>TIME</td>
<td>100.0</td>
<td>250.0</td>
<td>500.0</td>
<td>1000.0</td>
<td>2000.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>-------------</td>
<td>------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>--------</td>
<td>--------</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.01</td>
<td>8.7971</td>
<td>8.3515</td>
<td>6.7943</td>
<td>3.4626</td>
<td>1.7828</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>7.8389</td>
<td>7.3941</td>
<td>3.6702</td>
<td>1.9383</td>
<td>1.0692</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>6.0504</td>
<td>3.5375</td>
<td>3.0240</td>
<td>1.2620</td>
<td>0.8810</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>4.8577</td>
<td>2.7457</td>
<td>1.8731</td>
<td>1.4366</td>
<td>1.2183</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>4.5947</td>
<td>3.0952</td>
<td>2.5477</td>
<td>2.2738</td>
<td>2.1369</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>5.1154</td>
<td>3.8783</td>
<td>3.4392</td>
<td>3.2196</td>
<td>3.1098</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>6.7170</td>
<td>5.7048</td>
<td>5.3524</td>
<td>5.1762</td>
<td>5.0881</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>9.0138</td>
<td>8.1180</td>
<td>7.8090</td>
<td>7.6545</td>
<td>7.5773</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>11.4114</td>
<td>10.5746</td>
<td>10.2873</td>
<td>10.1437</td>
<td>10.0718</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1.50</td>
<td>16.3084</td>
<td>15.5312</td>
<td>15.2656</td>
<td>15.1328</td>
<td>15.0664</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.00</td>
<td>21.2568</td>
<td>20.5095</td>
<td>20.2548</td>
<td>20.1274</td>
<td>20.0637</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>31.2050</td>
<td>30.4878</td>
<td>30.2439</td>
<td>30.1220</td>
<td>30.0610</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td>41.1791</td>
<td>40.4770</td>
<td>40.2385</td>
<td>40.1192</td>
<td>40.0596</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>TIME</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>14.7591</td>
<td>11.9932</td>
<td>8.3869</td>
<td>4.4626</td>
<td>2.2828</td>
<td></td>
</tr>
<tr>
<td>0.02</td>
<td>15.6505</td>
<td>10.3546</td>
<td>5.6701</td>
<td>2.9383</td>
<td>1.5692</td>
<td></td>
</tr>
<tr>
<td>0.05</td>
<td>15.3759</td>
<td>7.5369</td>
<td>4.0240</td>
<td>2.2620</td>
<td>1.3810</td>
<td></td>
</tr>
<tr>
<td>0.10</td>
<td>14.5783</td>
<td>6.7456</td>
<td>3.8731</td>
<td>2.4366</td>
<td>1.7183</td>
<td></td>
</tr>
<tr>
<td>0.20</td>
<td>14.4352</td>
<td>7.0952</td>
<td>4.5477</td>
<td>3.2738</td>
<td>2.6369</td>
<td></td>
</tr>
<tr>
<td>0.30</td>
<td>14.9839</td>
<td>7.8783</td>
<td>5.4392</td>
<td>4.2196</td>
<td>3.6098</td>
<td></td>
</tr>
<tr>
<td>0.50</td>
<td>16.6042</td>
<td>9.7048</td>
<td>7.3524</td>
<td>6.1762</td>
<td>5.5881</td>
<td></td>
</tr>
<tr>
<td>0.75</td>
<td>18.9093</td>
<td>12.1180</td>
<td>9.8090</td>
<td>8.6545</td>
<td>8.0773</td>
<td></td>
</tr>
<tr>
<td>1.00</td>
<td>21.3107</td>
<td>14.5746</td>
<td>12.2873</td>
<td>11.1437</td>
<td>10.5718</td>
<td></td>
</tr>
<tr>
<td>1.50</td>
<td>26.2115</td>
<td>19.5312</td>
<td>17.2656</td>
<td>16.1328</td>
<td>15.5664</td>
<td></td>
</tr>
<tr>
<td>2.00</td>
<td>31.1616</td>
<td>24.5095</td>
<td>22.2548</td>
<td>21.1274</td>
<td>20.5637</td>
<td></td>
</tr>
<tr>
<td>3.00</td>
<td>41.1116</td>
<td>34.4878</td>
<td>32.2439</td>
<td>31.1220</td>
<td>30.5610</td>
<td></td>
</tr>
<tr>
<td>4.00</td>
<td>51.0865</td>
<td>44.4770</td>
<td>42.2385</td>
<td>41.1192</td>
<td>40.5596</td>
<td></td>
</tr>
</tbody>
</table>

A-61
<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>81.1085</td>
<td>78.9736</td>
<td>61.0500</td>
<td>32.7265</td>
<td>16.4276</td>
</tr>
<tr>
<td>0.02</td>
<td>68.7773</td>
<td>58.1803</td>
<td>32.9020</td>
<td>16.5832</td>
<td>8.3916</td>
</tr>
<tr>
<td>0.05</td>
<td>46.6783</td>
<td>26.8755</td>
<td>13.7398</td>
<td>7.1199</td>
<td>3.8100</td>
</tr>
<tr>
<td>0.10</td>
<td>29.8565</td>
<td>14.4571</td>
<td>7.7310</td>
<td>4.3655</td>
<td>2.6828</td>
</tr>
<tr>
<td>0.20</td>
<td>18.1060</td>
<td>8.9525</td>
<td>5.4766</td>
<td>3.7383</td>
<td>2.8692</td>
</tr>
<tr>
<td>0.30</td>
<td>14.2857</td>
<td>7.7833</td>
<td>5.3918</td>
<td>4.1959</td>
<td>3.5980</td>
</tr>
<tr>
<td>0.50</td>
<td>12.2831</td>
<td>8.0478</td>
<td>6.5240</td>
<td>5.7620</td>
<td>5.3810</td>
</tr>
<tr>
<td>0.75</td>
<td>12.7429</td>
<td>9.6800</td>
<td>8.5901</td>
<td>8.0450</td>
<td>7.7725</td>
</tr>
<tr>
<td>1.00</td>
<td>14.2146</td>
<td>11.7461</td>
<td>10.8731</td>
<td>10.4366</td>
<td>10.2183</td>
</tr>
<tr>
<td>1.50</td>
<td>18.1813</td>
<td>16.3122</td>
<td>15.6561</td>
<td>15.3281</td>
<td>15.1640</td>
</tr>
<tr>
<td>2.00</td>
<td>22.6629</td>
<td>21.0953</td>
<td>20.5477</td>
<td>20.2738</td>
<td>20.1369</td>
</tr>
<tr>
<td>3.00</td>
<td>32.1434</td>
<td>30.8783</td>
<td>30.4392</td>
<td>30.2196</td>
<td>30.1098</td>
</tr>
<tr>
<td>4.00</td>
<td>41.8832</td>
<td>40.7699</td>
<td>40.3849</td>
<td>40.1925</td>
<td>40.0962</td>
</tr>
</tbody>
</table>

**CS1=10, CS2=100, AND CS3=100**

<table>
<thead>
<tr>
<th>REPAIR RATE</th>
<th>100.0</th>
<th>250.0</th>
<th>500.0</th>
<th>1000.0</th>
<th>2000.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>9.6971</td>
<td>9.2515</td>
<td>7.2943</td>
<td>4.3626</td>
<td>2.6828</td>
</tr>
<tr>
<td>0.02</td>
<td>9.6389</td>
<td>8.1941</td>
<td>5.4702</td>
<td>3.7383</td>
<td>2.8692</td>
</tr>
<tr>
<td>0.05</td>
<td>10.5504</td>
<td>8.0375</td>
<td>6.5240</td>
<td>5.7620</td>
<td>5.3810</td>
</tr>
<tr>
<td>0.10</td>
<td>13.8577</td>
<td>11.7457</td>
<td>10.8731</td>
<td>10.4366</td>
<td>10.2183</td>
</tr>
<tr>
<td>0.20</td>
<td>22.5947</td>
<td>21.0952</td>
<td>20.5477</td>
<td>20.2738</td>
<td>20.1369</td>
</tr>
<tr>
<td>0.30</td>
<td>32.1154</td>
<td>30.8783</td>
<td>30.4392</td>
<td>30.2196</td>
<td>30.1098</td>
</tr>
<tr>
<td>0.50</td>
<td>51.7170</td>
<td>50.7048</td>
<td>50.3524</td>
<td>50.1762</td>
<td>50.0881</td>
</tr>
<tr>
<td>0.75</td>
<td>76.5138</td>
<td>75.6180</td>
<td>75.3090</td>
<td>75.1545</td>
<td>75.0773</td>
</tr>
<tr>
<td>1.00</td>
<td>101.4114</td>
<td>100.5746</td>
<td>100.2873</td>
<td>100.1437</td>
<td>100.0718</td>
</tr>
<tr>
<td>1.50</td>
<td>151.3084</td>
<td>150.5312</td>
<td>150.2656</td>
<td>150.1328</td>
<td>150.0664</td>
</tr>
<tr>
<td>2.00</td>
<td>201.2568</td>
<td>200.5095</td>
<td>200.2548</td>
<td>200.1274</td>
<td>200.0637</td>
</tr>
<tr>
<td>3.00</td>
<td>301.2050</td>
<td>300.4878</td>
<td>300.2439</td>
<td>300.1220</td>
<td>300.0610</td>
</tr>
<tr>
<td>4.00</td>
<td>401.1791</td>
<td>400.4770</td>
<td>400.2385</td>
<td>400.1192</td>
<td>400.0596</td>
</tr>
</tbody>
</table>
Figure A3.6 shows how the average cost changes with $\mu$ for the four sets of cost factors. The minimum for these plots occurs at different values of $\mu$ due to changes in the cost factors. In Figure A3.7, the average costs are shown for various time horizons. We note that all the curves follow a similar pattern, i.e., first decreasing with $\mu$ and then increasing.
Figure A3.5. Average Availability vs. Repair Rate for Different Failure Rates ($t = 500$).
Figure A3.6. Expected Total Cost/Unit Time vs. Repair Rate for Different Cost Factors ($\lambda = .05, t = 500$).
Figure A3.7. Expected Total Cost/Unit Time vs. Repair Rate for Different Times ($\lambda = .05$, $c_{s_1} = 10$, $c_{s_2} = 10$, $c_{s_3} = 10$).

A-66
A3.3 Operational Cost Model: Hardware-Software System

We consider a hardware-software system whose behavior is the same as the system discussed in Section A2.1. Having discussed the cost models for hardware only and software only systems in the previous Sections, the operational cost of the hardware-software system is basically the sum of both the operational costs. However, the performance measures required in the hardware-software system are not obviously equal to their respective sums.

The cost elements associated with the operation of this system are $c_{h_1}$, $c_{h_2}$, and $c_{h_3}$ as defined in Section A3.1, and $c_{s_1}$, $c_{s_2}$, and $c_{s_3}$ as defined in Section A3.2.

The performance measures required for the cost model are: $M_h(t)$ and $M_s(t)$, the expected number of hardware and software failures by time $t$, respectively, and the expected total down time during $(0,t)$.

Let $C(t)$ be the expected total operational cost associated with the hardware-software system and let $c_{h_3} = c_{s_3} = c_3$. Then

$$C(t) = c_{h_1} M_h(t) + c_{s_1} M_s(t) + c_{h_2} \gamma t + c_{s_2} \mu t$$

$$+ c_3 \int_0^t (1 - A(x)) dx,$$

(A3.15)
where the expressions for $M_h(t)$, $M_s(t)$, and $A(\cdot)$ are given in Equations (A2.49), (A2.54), and (A2.43), respectively.
Illustrative Examples

Now we numerically study the behavior of $M_h(t)$, $M_s(t)$, $C(t)/t$ and $A_{av}(t)$ as a function of $y$ and $\mu$. The values of $A(x)$, $A_{av}(t)$, $M_s(t)$ and $M_h(t)$ are computed from Equations (A2.43), (A2.44), (A2.45), and (A2.49), respectively. The values of $C(t)$ are given by Equation (A3.15).

Let us consider a system with $N = 10$, $p_s = .9$, $p_h = .9$, $\beta = .01$, and $\lambda = .05$. For $t = 100$, $y = .02$ to 1.0 and $\mu = 0.01$ to 0.50, the values of $A_{av}(t)$, $M_s(t)$ and $M_h(t)$ are given in Table A3.5. We note that the average availability improves with both the hardware and the software repair rates. Also, the expected number of failures increases with increase in $y$ and $\mu$. This is because of the increased amount of time that the system is up leading to a longer time available for the failures to occur. Note that for these data sets all software errors have been removed by approximately $t = 500$. As pointed out earlier, after this happens, the system behaves as a hardware only system. To see how $C(t)/t$ behaves as a function of $y$ and $\mu$, let us suppose that $c_{s_1} = 10$, $c_{h_1} = 10$, $c_{s_2} = 10$, $c_{h_2} = 10$, $c_{s_3} = c_{h_3} = 10$, and $t = 100$ to 2000 as shown in Table A3.9. As can be easily seen, the cost varies with both $y$ and $\mu$. As an example, for $t = 500$, $\mu = 0.10$, $C(t)/t$ goes from 5.97 to 12.23 as $y$ goes from 0.02 to 1.00. The
minimum seems to occur around $\gamma = 0.10$. Similarly, for $\gamma = 0.10$, $t = 500$, $C_s(t)$ goes from 9.07 to 7.63 as $\mu$ goes from 0.01 to 0.50 with the minimum occurring at around $\mu = 0.10$. A similar behavior is seen for other $t$ values.
### TABLE A3.5

**AVERAGE AVAILABILITY AND EXPECTED NUMBER OF FAILURES, HARDWARE-SOFTWARE SYSTEM**

(\( t = 100 \))

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.200</th>
<th>0.500</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>0.1814651</td>
<td>0.1866103</td>
<td>0.1897074</td>
<td>0.1916580</td>
<td>0.1929772</td>
<td>0.1934441</td>
</tr>
<tr>
<td>0.050</td>
<td>0.4625249</td>
<td>0.4938450</td>
<td>0.5143947</td>
<td>0.5283556</td>
<td>0.5383869</td>
<td>0.5420751</td>
</tr>
<tr>
<td>0.100</td>
<td>0.5831032</td>
<td>0.6329669</td>
<td>0.6664243</td>
<td>0.6896547</td>
<td>0.7066430</td>
<td>0.7129479</td>
</tr>
<tr>
<td>0.250</td>
<td>0.6842970</td>
<td>0.7542586</td>
<td>0.8014733</td>
<td>0.8342371</td>
<td>0.8580026</td>
<td>0.8667412</td>
</tr>
<tr>
<td>0.500</td>
<td>0.7216969</td>
<td>0.8003935</td>
<td>0.8532790</td>
<td>0.8897114</td>
<td>0.9159166</td>
<td>0.9254982</td>
</tr>
</tbody>
</table>

**EXPECTED NUMBER OF SOFTWARE FAILURES**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>9.5311</td>
<td>9.9464</td>
<td>10.1424</td>
<td>10.2383</td>
<td>10.2913</td>
<td>10.3080</td>
</tr>
<tr>
<td>0.050</td>
<td>9.9296</td>
<td>10.4010</td>
<td>10.6042</td>
<td>10.6928</td>
<td>10.7368</td>
<td>10.7498</td>
</tr>
<tr>
<td>0.100</td>
<td>10.1058</td>
<td>10.6000</td>
<td>10.7944</td>
<td>10.8697</td>
<td>10.9036</td>
<td>10.9131</td>
</tr>
<tr>
<td>0.250</td>
<td>10.1547</td>
<td>10.6534</td>
<td>10.8423</td>
<td>10.9121</td>
<td>10.9423</td>
<td>10.9506</td>
</tr>
<tr>
<td>0.500</td>
<td>10.1840</td>
<td>10.1892</td>
<td>10.1923</td>
<td>10.1943</td>
<td>10.1956</td>
<td>0.1961</td>
</tr>
</tbody>
</table>

**EXPECTED NUMBER OF HARDWARE FAILURES**

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.1840</th>
<th>0.1892</th>
<th>0.1923</th>
<th>0.1943</th>
<th>0.1956</th>
<th>0.1961</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>0.4650</td>
<td>0.4965</td>
<td>0.5170</td>
<td>0.5310</td>
<td>0.5410</td>
<td>0.5447</td>
</tr>
<tr>
<td>0.050</td>
<td>0.5856</td>
<td>0.6356</td>
<td>0.6690</td>
<td>0.6923</td>
<td>0.7093</td>
<td>0.7156</td>
</tr>
<tr>
<td>0.100</td>
<td>0.6868</td>
<td>0.7569</td>
<td>0.8041</td>
<td>0.8369</td>
<td>0.8606</td>
<td>0.8694</td>
</tr>
<tr>
<td>0.250</td>
<td>0.7242</td>
<td>0.3030</td>
<td>0.8559</td>
<td>0.8923</td>
<td>0.9185</td>
<td>0.9281</td>
</tr>
</tbody>
</table>
### TABLE A3.6

**AVERAGE AVAILABILITY AND EXPECTED NUMBER OF FAILURES, HARDWARE-SOFTWARE SYSTEM**

\( (t = 500) \)

<table>
<thead>
<tr>
<th>Software Repair Rate</th>
<th>Average Availability</th>
<th>Hardware Repair Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.020</td>
<td>0.050</td>
</tr>
<tr>
<td>0.010</td>
<td>0.2838317</td>
<td>0.3315687</td>
</tr>
<tr>
<td>0.050</td>
<td>0.5842655</td>
<td>0.7177649</td>
</tr>
<tr>
<td>0.100</td>
<td>0.6260339</td>
<td>0.7710168</td>
</tr>
<tr>
<td>0.250</td>
<td>0.6511171</td>
<td>0.8029690</td>
</tr>
<tr>
<td>0.500</td>
<td>0.6594803</td>
<td>0.8136197</td>
</tr>
</tbody>
</table>

**Expected Number of Software Failures**

<table>
<thead>
<tr>
<th>Software Repair Rate</th>
<th>0.010</th>
<th>0.050</th>
<th>0.100</th>
<th>0.250</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.050</td>
<td>11.1041</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.100</td>
<td>11.1060</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.250</td>
<td>11.1069</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.500</td>
<td>11.1071</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
</tbody>
</table>

**Expected Number of Hardware Failures**

<table>
<thead>
<tr>
<th>Software Repair Rate</th>
<th>0.010</th>
<th>0.050</th>
<th>0.100</th>
<th>0.250</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1.4218</td>
<td>1.6605</td>
<td>1.7893</td>
<td>1.8695</td>
<td>1.9241</td>
</tr>
<tr>
<td>0.050</td>
<td>2.9239</td>
<td>3.5914</td>
<td>3.9242</td>
<td>4.1230</td>
<td>4.2550</td>
</tr>
<tr>
<td>0.100</td>
<td>3.1328</td>
<td>3.8577</td>
<td>4.2171</td>
<td>4.4313</td>
<td>4.5734</td>
</tr>
<tr>
<td>0.250</td>
<td>3.2582</td>
<td>4.0175</td>
<td>4.3928</td>
<td>4.6163</td>
<td>4.7644</td>
</tr>
<tr>
<td>0.500</td>
<td>3.3000</td>
<td>4.0707</td>
<td>4.4514</td>
<td>4.6779</td>
<td>4.8281</td>
</tr>
<tr>
<td>SOFTWARE REPAIR RATE</td>
<td>0.020</td>
<td>0.050</td>
<td>0.100</td>
<td>0.200</td>
<td>0.500</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>0.010</td>
<td>0.4464477</td>
<td>0.5552353</td>
<td>0.6081232</td>
<td>0.6393556</td>
<td>0.6599666</td>
</tr>
<tr>
<td>0.050</td>
<td>0.6135084</td>
<td>0.7679722</td>
<td>0.8421592</td>
<td>0.8857214</td>
<td>0.9143735</td>
</tr>
<tr>
<td>0.100</td>
<td>0.6344296</td>
<td>0.7945992</td>
<td>0.8714489</td>
<td>0.9165528</td>
<td>0.9462101</td>
</tr>
<tr>
<td>0.250</td>
<td>0.6469823</td>
<td>0.8105754</td>
<td>0.8890227</td>
<td>0.9350515</td>
<td>0.9653120</td>
</tr>
<tr>
<td>0.500</td>
<td>0.6511665</td>
<td>0.8159008</td>
<td>0.8948806</td>
<td>0.9412177</td>
<td>0.9716794</td>
</tr>
</tbody>
</table>

**AVERAGE AVAILABILITY**

**EXPECTED NUMBER OF SOFTWARE FAILURES**

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.200</th>
<th>0.500</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>11.1100</td>
<td>11.1106</td>
<td>11.1107</td>
<td>11.1107</td>
<td>11.1108</td>
<td>11.1108</td>
</tr>
<tr>
<td>0.050</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.100</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.250</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
<tr>
<td>0.500</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
<td>11.1111</td>
</tr>
</tbody>
</table>

**EXPECTED NUMBER OF HARDWARE FAILURES**

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.200</th>
<th>0.500</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>4.4671</td>
<td>5.5550</td>
<td>6.0838</td>
<td>6.3962</td>
<td>6.6023</td>
<td>6.6745</td>
</tr>
<tr>
<td>0.050</td>
<td>6.1377</td>
<td>7.6823</td>
<td>8.4242</td>
<td>8.8598</td>
<td>9.1463</td>
<td>9.2465</td>
</tr>
<tr>
<td>0.100</td>
<td>6.3469</td>
<td>7.9486</td>
<td>8.7171</td>
<td>9.1681</td>
<td>9.4647</td>
<td>9.5684</td>
</tr>
</tbody>
</table>
### Table A3.8

**Average Availability and Expected Number of Failures, Hardware-Software System**

(t = 2000)

<table>
<thead>
<tr>
<th>Software Repair Rate</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.200</th>
<th>0.500</th>
<th>1.000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td>0.544</td>
<td>0.686</td>
<td>0.753</td>
<td>0.793</td>
<td>0.819</td>
<td>0.828</td>
</tr>
<tr>
<td>0.050</td>
<td>0.628</td>
<td>0.793</td>
<td>0.871</td>
<td>0.916</td>
<td>0.946</td>
<td>0.957</td>
</tr>
<tr>
<td>0.100</td>
<td>0.637</td>
<td>0.806</td>
<td>0.885</td>
<td>0.932</td>
<td>0.962</td>
<td>0.973</td>
</tr>
<tr>
<td>0.250</td>
<td>0.645</td>
<td>0.814</td>
<td>0.895</td>
<td>0.941</td>
<td>0.972</td>
<td>0.983</td>
</tr>
<tr>
<td>0.500</td>
<td>0.647</td>
<td>0.817</td>
<td>0.897</td>
<td>0.944</td>
<td>0.975</td>
<td>0.986</td>
</tr>
</tbody>
</table>

**Expected Number of Software Failures**

<table>
<thead>
<tr>
<th>Software Repair Rate</th>
<th>11.111</th>
<th>11.111</th>
<th>11.111</th>
<th>11.111</th>
<th>11.111</th>
<th>11.111</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Expected Number of Hardware Failures**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0.010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.050</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.100</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.250</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0.500</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### TABLE A3.9

**EXPECTED TOTAL COST PER UNIT TIME**

**HARDWARE-SOFTWARE SYSTEM**

\[ CS_1=10, \ CH_1=10, \ CS_2=10, \ CH_2=10, \ CS_3=CH_3=10 \]

\( t = 100 \)

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.50</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>9.13</td>
<td>9.40</td>
<td>9.87</td>
<td>10.86</td>
<td>13.85</td>
<td>18.84</td>
</tr>
<tr>
<td>0.05</td>
<td>7.07</td>
<td>7.11</td>
<td>7.42</td>
<td>8.29</td>
<td>11.20</td>
<td>16.16</td>
</tr>
<tr>
<td>0.10</td>
<td>6.42</td>
<td>6.27</td>
<td>6.46</td>
<td>7.24</td>
<td>10.08</td>
<td>15.02</td>
</tr>
<tr>
<td>0.25</td>
<td>6.94</td>
<td>6.59</td>
<td>6.65</td>
<td>7.33</td>
<td>10.10</td>
<td>15.01</td>
</tr>
<tr>
<td>0.50</td>
<td>9.07</td>
<td>8.64</td>
<td>8.64</td>
<td>9.28</td>
<td>12.03</td>
<td>16.93</td>
</tr>
</tbody>
</table>

\( t = 500 \)

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.50</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>9.14</td>
<td>8.88</td>
<td>9.07</td>
<td>9.88</td>
<td>12.75</td>
<td>17.70</td>
</tr>
<tr>
<td>0.05</td>
<td>5.97</td>
<td>4.68</td>
<td>4.39</td>
<td>4.92</td>
<td>7.60</td>
<td>12.49</td>
</tr>
<tr>
<td>0.10</td>
<td>5.97</td>
<td>4.55</td>
<td>4.19</td>
<td>4.68</td>
<td>7.34</td>
<td>12.23</td>
</tr>
<tr>
<td>0.25</td>
<td>7.17</td>
<td>5.67</td>
<td>5.27</td>
<td>5.74</td>
<td>8.39</td>
<td>13.27</td>
</tr>
<tr>
<td>0.50</td>
<td>9.57</td>
<td>8.04</td>
<td>7.63</td>
<td>8.09</td>
<td>10.74</td>
<td>15.61</td>
</tr>
</tbody>
</table>

\( t = 1000 \)

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.50</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>9.31</td>
<td>7.88</td>
<td>7.54</td>
<td>8.05</td>
<td>10.72</td>
<td>15.60</td>
</tr>
<tr>
<td>0.05</td>
<td>7.06</td>
<td>4.90</td>
<td>4.22</td>
<td>4.53</td>
<td>7.07</td>
<td>11.91</td>
</tr>
<tr>
<td>0.10</td>
<td>7.22</td>
<td>4.98</td>
<td>4.26</td>
<td>4.54</td>
<td>7.07</td>
<td>11.90</td>
</tr>
<tr>
<td>0.25</td>
<td>8.52</td>
<td>6.22</td>
<td>5.48</td>
<td>5.74</td>
<td>8.26</td>
<td>13.09</td>
</tr>
<tr>
<td>0.50</td>
<td>10.96</td>
<td>8.64</td>
<td>7.88</td>
<td>8.15</td>
<td>10.66</td>
<td>15.49</td>
</tr>
</tbody>
</table>

\( t = 2000 \)

<table>
<thead>
<tr>
<th>SOFTWARE REPAIR RATE</th>
<th>0.02</th>
<th>0.05</th>
<th>0.10</th>
<th>0.20</th>
<th>0.50</th>
<th>1.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.01</td>
<td>8.61</td>
<td>6.37</td>
<td>5.66</td>
<td>5.96</td>
<td>8.50</td>
<td>13.34</td>
</tr>
<tr>
<td>0.05</td>
<td>7.51</td>
<td>4.86</td>
<td>3.96</td>
<td>4.15</td>
<td>6.62</td>
<td>11.43</td>
</tr>
<tr>
<td>0.10</td>
<td>7.82</td>
<td>5.12</td>
<td>4.20</td>
<td>4.37</td>
<td>6.83</td>
<td>11.64</td>
</tr>
<tr>
<td>0.25</td>
<td>9.21</td>
<td>6.48</td>
<td>5.54</td>
<td>5.71</td>
<td>8.16</td>
<td>12.97</td>
</tr>
<tr>
<td>0.50</td>
<td>11.67</td>
<td>8.93</td>
<td>7.99</td>
<td>8.15</td>
<td>10.60</td>
<td>15.41</td>
</tr>
</tbody>
</table>

A-75
Figure A3.8. Contours of Average Availability vs. Repair Rates: Hardware-Software System ($\beta = .01$, $\lambda = .05$, $t = 500$).
Figure A3.9. Surface of Average Availability vs. Repair Rates: Hardware-Software System ($\beta = .01$, $\lambda = .05$, $t = 500$).
Figure A3.10. Contours of Expected Total Cost/Unit Time vs. Repair Rates: Hardware-Software System ($\beta = .01$, $\lambda = .05$, $t = 500$, cost factors = 10).
Figure A3.11. Expected Total Cost/Unit Time vs. Repair Rates: Hardware-Software System ($\beta = .01$, $\lambda = .05$, $t = 500$).