Chap. 8 Reliability Growth Models

1. Introduction
2. The Exponential Model
3. Reliability Growth Models
4. Modeling Process
5. Identifying Trends in Failure Data

Appendix: Laplace Trend Test

1. Introduction

Software Reliability Models

-Software reliability models are statistical models which can be used to make predictions about a software system's failure rate, given the failure history of the system.

-The models make assumptions about the fault discovery and removal process. These assumptions determine the form of the model and the meaning of the model's parameters.

Notions of Reliability Growth

-In contrast to Rayleigh, which models the defect pattern of the entire development process, reliability growth models are usually based on data from the formal testing phases.

- In practice, such models are applied during the final testing phase when the development is virtually completed.

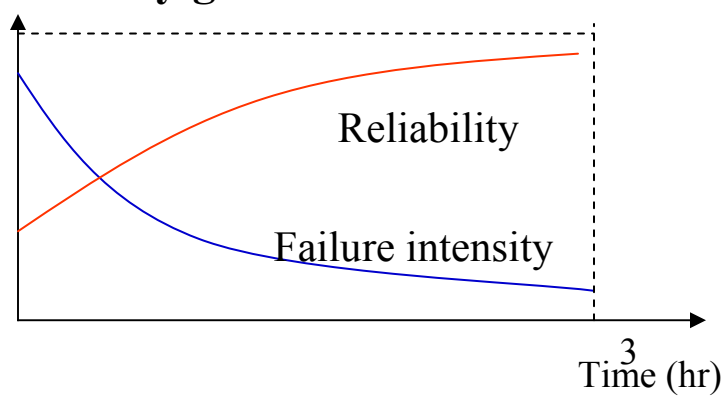
- The rationale is that defect arrival or failure patterns during such testing are good indicators of the product's reliability when it is used by customers.

- During such post-development testing, when failures occur and defects are identified and fixed, the software becomes more stable, and reliability grows over time.

- Hence models that address such a process are called **reliability growth models**.

-There are two types of models:

- those that predict ***times between failures***, and
- those that predict the ***number of failures*** that will be found in future test intervals.



Time between failures models

-Models that predict times between failures can be expressed as a probability density function, $f_i(t)$ whose parameters are estimated based on the values of previously observed times between failures $t_1, t_2, \ldots, t_{i-1}$.

-This probability density function is used to predict the time to the next failure as well as the reliability of the software system.

-Suppose that we've observed **i-1** times between failures since the start of testing, and we want to predict the time between failure **i-1** and failure **i**, which we'll represent by the random variable **t**.

- The expected value of **t** is what we're looking for, and this is given by:

$$E[t] = \int_0^{+\infty} t f_i(t)\, dt$$

where $f_i(t)$ is the probability density function representing the particular model we're using. The parameters of this probability density function can be estimated using, for instance, the **Maximum Likelihood** or **Least Squares** methods.

-Since $f_i(t)$ is used to predict the time to the next failure, we can use it to predict the reliability of the system.

- the probability that the time to the next failure will be less than a certain value, **x**, is simply:

$$P(t \leq x) = \int_0^x f_i(t)\,dt$$

-Software reliability is defined as the probability that a software system will run without failure for a specified time in a specified environment.

- Using this definition, then, the reliability of the software over an interval of time of length **x** is:

$Rel(x) = 1 - P(x)$; this implies:

$$Rel(x) = \int_x^{+\infty} f_i(t)\,dt$$

Failure Count Models

-For models that predict the number of failures in future test intervals, we also have a probability density function $f_i(t)$.

- The parameters of $f_i(t)$ are computed based on the failure counts in the previous $(i-1)$ test intervals.

-Suppose that we've observed failure counts in test intervals $f_1, f_2, \ldots, f_i$, and we want to predict what the number of failures will be in interval i+1.

- Representing this quantity by the random variable **x**, we'll get our prediction by finding the expected value of **x**:

$$E[x] = \int_0^{+\infty} x f_i(x) dx$$

2. The Exponential Model

-The exponential model is another special case of the Weibull family, with the shape parameter *m* equal to 1.

•It is best used for statistical processes that decline monotonically to an asymptote.

-Its CDF and PDF are

$$CDF: \quad F(t) = K\left[1 - e^{-\left(\frac{t}{c}\right)}\right] = K\left[1 - e^{-\lambda t}\right]$$

$$PDF: \quad f(t) = \frac{K}{c} e^{-\left(\frac{t}{c}\right)} = K \lambda e^{-\lambda t}$$

Where *c* is the scale parameter, *t* is time, and $\lambda = 1/c$ is the *error detection rate* or *instantaneous failure rate*. It is also referred to as the *hazard rate*. *K* represents the total number of defects or the total cumulative defect rate.

-**Note:** *K* and λ are the two parameters for estimation when deriving a specific model from a data set.

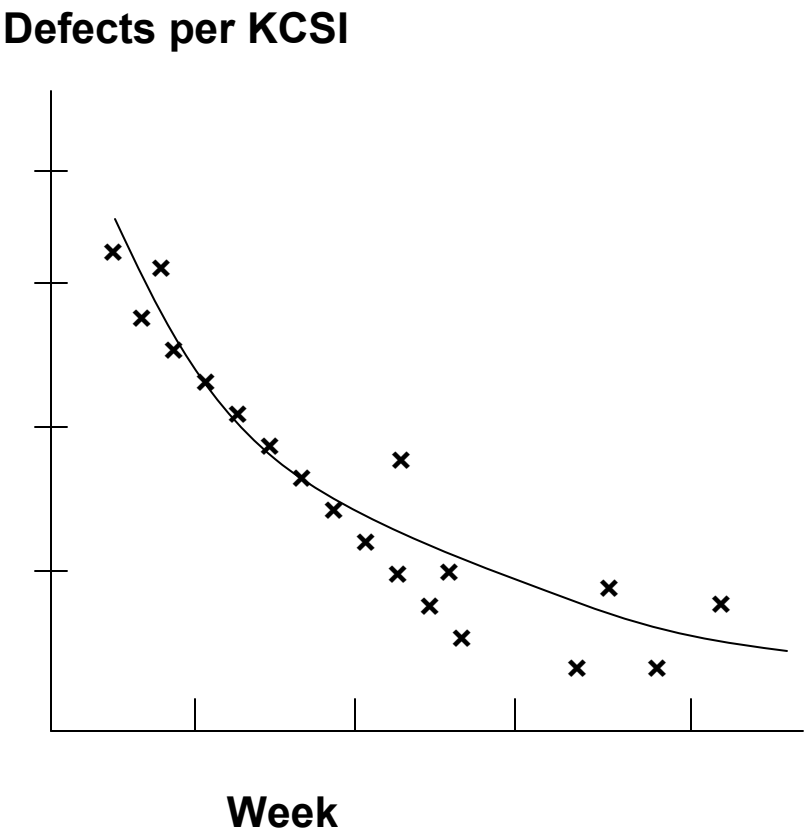
-The exponential distribution is the simplest and most important distribution in reliability analysis.

-It is one of the better known models and is often the basis of many other software reliability growth models.

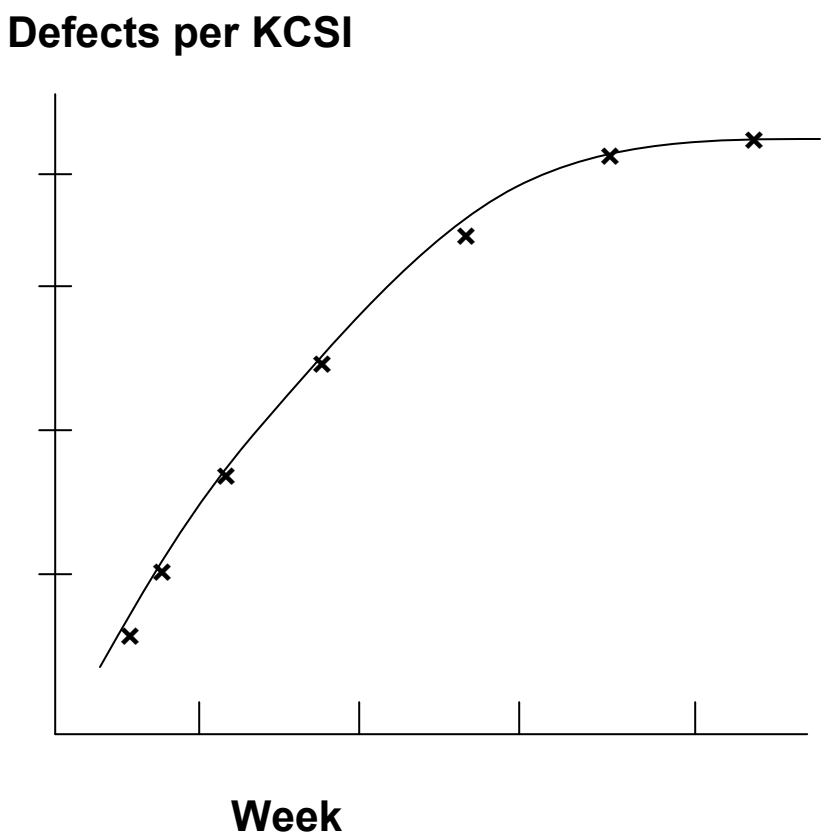
-When applying the exponential model for reliability analysis, data tracking is done either in terms of precise CPU execution time or on a calendar-time basis.

-In principle, execution-time tracking is for small projects while calendar-time is common for commercial development.

Exponential Model



Density Distribution



Cumulative Distribution

Example: for λ=0.001 or 1 failure for 1000 hours, reliability (R) is around 0.992 for 8 hours of operation.

4. Reliability Growth Models

-The exponential model can be regarded as the basic form of software reliability growth model.

- For the past decades, more than a hundred models have been proposed in the research literature.
- Unfortunately few have been tested in practical environments with real data, and even fewer are in use.

-Examples of models currently being used include the following:

Times Between Failures (TBF) Models	Failures Counts (FC) Models
Geometric	Generalized Poisson
Jelinski-Moranda	Generalized Poisson-User specified interval weighting
Littlewood-Verrall Linear	Nonhomogeneous Poisson (NHPP)
Littlewood-Verrall Quadratic	Schneidewind
Musa Basic	Schneidewind- ignore first "s-1" test intervals
Musa-Okumuto	Schneidewind – total failures in first "s-1" test intervals*
Nonhomogeneous Poisson (NHPP)	Shick-Wolverton*
	Yamada S-shaped

The Jelinski-Moranda Model

-The Jelinski-Moranda model is a time between failures model.

-This model makes the following assumptions about the fault detection and correction process:

a. The rate of fault detection is proportional to the current fault content of the program.
b. All failures are equally likely to occur and are independent of each other.
c. Each failure is of the same order of severity as any other failure.
d. The failure rate remains constant over the interval between failure occurrences.
e. During test, the software is operated in a similar manner as the expected operational usage.
f. The faults are corrected instantaneously without introduction of new faults into the program.

-From assumptions a, b, d, and f, we can write the **hazard rate** (the instantaneous failure rate) as:

$$Z(t) = \Phi \times (N - (i - 1))$$

- Where t is any time between the discovery of the i'th and (i-1)th failure; Φ is the proportionality constant given in assumption (a), and N is the total number of faults initially in the system.
- This means that if (i-1) faults has been discovered by time t, there would be N-(i-1) faults remaining in the system.

-If we represent the time between the ith and the $(i-1)^{th}$ failure by the random variable $\mathbf{X_i}$, from assumption (d) we can see that $\mathbf{X_i}$ has an exponential distribution, $\mathbf{f(X_i)}$, as shown below:

$$f(X_i) = \Phi \times (N - (i-1))e^{-\Phi(N-(i-1))X_i}$$

-Model parameters N and Φ can be estimated numerically using estimation methods like maximum likelihood or least squares.

Goel-Okumuto Imperfect Debugging Model

-The J-M model assumes perfect debugging; in practice this is not always the case. In the process of fixing a defect, new defects may be injected.

-Goel and Okumoto proposed an imperfect debugging model to overcome the limitation of the assumption.

- In this model, the hazard function is given by:

$$\boxed{Z(t_i) = \left[N - p(i-1)\right]\lambda}$$

-Where N is the number of faults at the start of testing, p is the probability of imperfect debugging, and λ is the failure rate per fault.

The Non-Homogeneous Poisson Process Model

-The Non-Homogeneous Poisson Process model is based on failure counts.

-This model makes the following assumptions about the fault detection and correction process:

a. During test, the software is operated in a similar manner as the anticipated operational usage.

b. The number of failures, $(f_1, f_2, \ldots, f_n)$ detected in each of the time intervals $[(0,t_1),(t_1,t_2),\ldots,(t_{n-1},t_n)]$ are independent for any finite collection of times $t_1<t_2<\ldots<t_n$.

c. Every fault has the same chance of being detected and is of the same severity as any other fault.

d. The cumulative number of faults detected at any time t, *N(t),* follows a Poisson distribution with mean $\mu(t)$. This mean is such that the expected number of failure occurrences for any time $(t, t+\Delta t)$ is proportional to the expected number of undetected faults at time *t*.

e. The expected cumulative number of faults function, $\mu(t)$, is assumed to be a bounded, non-decreasing function of *t* with: $\mu(t) = 0$ for $t=0$, and $\mu(t)=\nu_0$ for $t\rightarrow\infty$. where $\mathbf{\nu_0}$ is the expected total number of faults to be eventually detected in the testing process.

f. Errors are removed from the software without inserting new errors.

-From assumptions (d) and (e), for any time period (t,t+Δt),

$$\mu(t, t+\Delta t) - \mu(t) = b(\nu_0 - \mu(t))\Delta t + O(\Delta t)$$

- where **b** is the constant of proportionality and $O(\Delta t)/\Delta t = 0$ as $\Delta t \rightarrow 0$.
- we can pose $b=\lambda_0/\nu_0$, where λ_0 is the initial failure intensity at the start of the execution.

-As $\Delta t \rightarrow 0$, the mean function $\mu(t)$ satisfies the following differential equation:

$$\frac{d\mu(t)}{dt} = 1 - \frac{\lambda_0}{\nu_0}\mu(t)$$

-Under the initial condition $\mu(0) = 0$, the mean function is:

$$\boxed{\mu(t) = \nu_0 (1 - e^{-\frac{\lambda_0}{\nu_0}t})}$$

-From assumption (d), the probability that the cumulative number of failures, *N(t),* is less than *n* is:

$$\boxed{p(N(t) \leq n) = \frac{(\mu(t))^n}{n!} e^{-\mu(t)}}$$

- At this point, we can compute estimates for the model parameters, λ_0 and ν_0, using the maximum likelihood or least squares method.

-In the basic execution model, the mean failures experienced μ is expressed in terms of the execution time τ as

$$\mu(\tau) = \nu_0 \times \left(1 - e^{-\frac{\lambda_0}{\nu_0}\tau}\right)$$

Where:

-λ_0 stands for the initial failure intensity at the start of the execution.

-ν_0 stands for the total number of failures occurring over an infinite time period; it corresponds to the expected number of failures to be observed eventually.

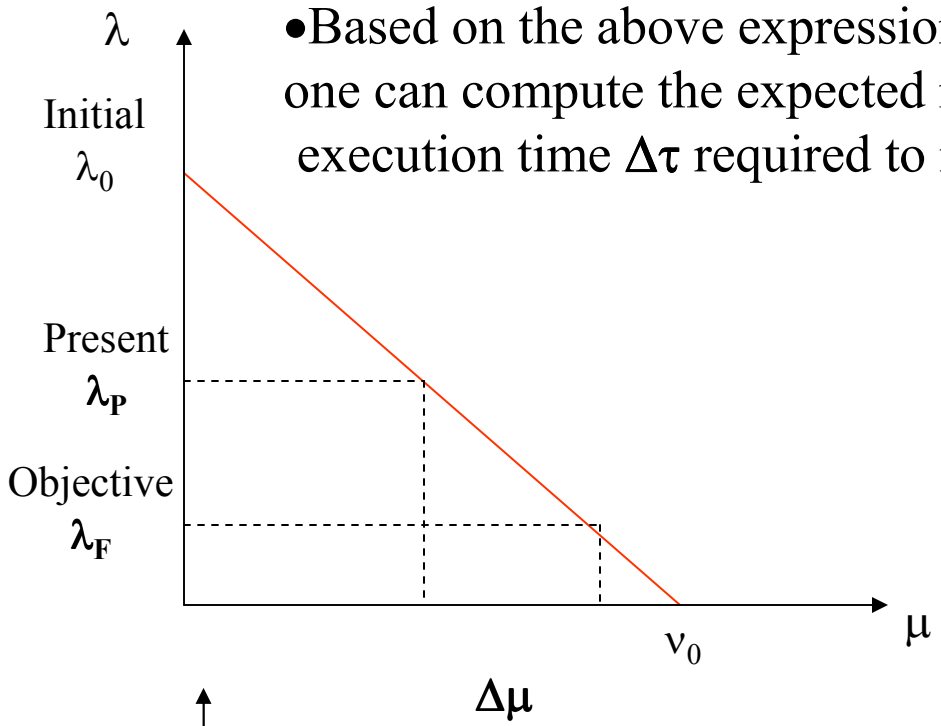
-The failure intensity expressed as a function of the execution time is given by

$$\lambda(\tau) = \lambda_0 \times e^{-\frac{\lambda_0}{\nu_0}\tau}$$

-Based on the above formula, the failure intensity λ is expressed in terms of μ as:

$$\lambda(\mu) = \lambda_0 \times \left(1 - \frac{\mu}{\nu_0}\right)$$

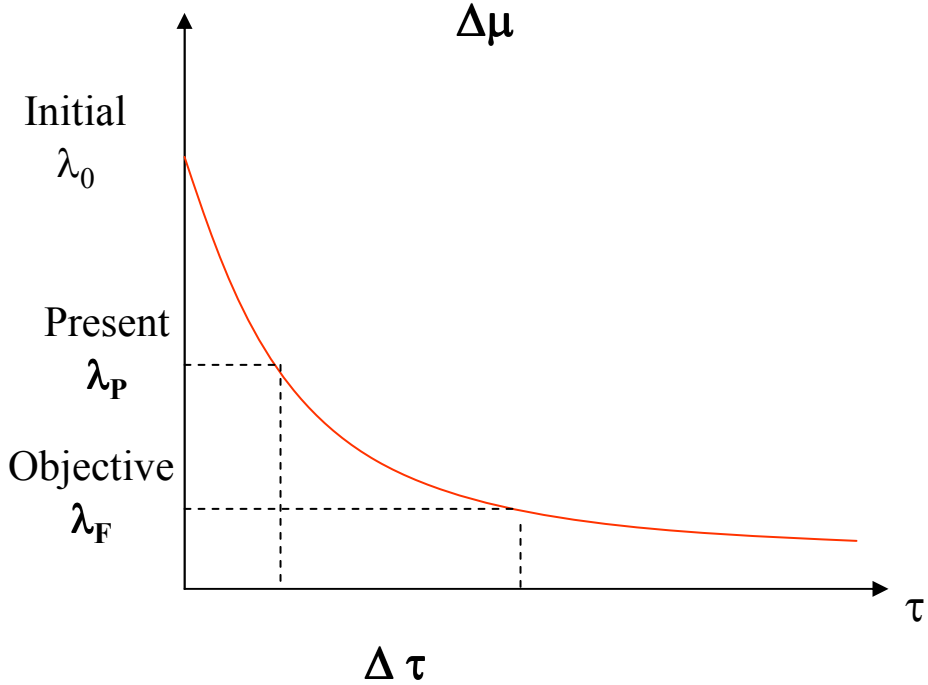
- Based on the above expressions, given some failure intensity objective, one can compute the expected number of failures $\Delta\mu$ and the additional execution time $\Delta\tau$ required to reach that objective.



$$\Delta\mu = \frac{\nu_0}{\lambda_0} \times (\lambda_1 - \lambda_2)$$

Where:

- λ_1 is the current failure intensity
- λ_2 is the failure intensity objective



$$\Delta\tau = \frac{\nu_0}{\lambda_0} \times \ln\left(\frac{\lambda_1}{\lambda_2}\right)$$

Example 7.8.1 : Assume that a program will experience 100 failures in infinite time. It has now experienced 50 failures. The failure intensity was 10 failures/CPU hr.

1. Calculate the current failure intensity.

2. Calculate the number of failures experienced after 10 and 100 CPU hr of execution.

3. Calculate the failure intensities at 10 and 100 CPU hr of execution.

4. Calculate the expected number of failures that will be experienced and the execution time between a current failure intensity of 3.68 failures/CPU hr and an objective of 0.000454 failure/CPU hr.

Musa-Okumuto Logarithmic Poisson Execution Time Model

-Concerned with modeling the number of failures observed in given testing intervals.

- Consider that the cumulative number of failures observed at time τ, $N(\tau)$, can be modeled as a non-homogeneous Poisson process (NHPP)-as a Poisson process with a time-dependent failure rate.

$$P\left(N(\tau)=n\right)=\frac{\left[\mu(\tau)\right]^{n}}{n!}e^{-\mu(\tau)}, \qquad n=0,1,2,\ldots$$

-Where $\mu(\tau)$ (mean value function), the expected number of failures observed by time τ :

$$\mu(\tau)=\frac{1}{\theta}\ln(\lambda_0\theta\tau+1)$$

-Where λ_0 is the initial failure intensity, and θ is the rate of reduction in the normalized failure intensity per failure (also referred to as the failure intensity decay).

- It attempts to consider that later fixes have a smaller effect on software's reliability than earlier ones.
- It is claimed to be superior for highly nonuniform operational user profiles, where some functions are executed much more frequently than others.

The Delayed S and Inflection S Models

-The rationale behind this model is that a testing process consists of not only a defect detection process, but also a defect isolation process.

-Because of the time needed for failure analysis, significant delay can occur between the time of the first failure observation and the time of reporting.

-In the delayed S-shaped reliability growth model, the cumulative number of detected defects is represented by an S-shaped curve.

-The model is based on the nonhomogeneous Poisson process, but with a different mean value function to reflect the delay in failure reporting:

$$\mu(t) = K\left[1-(1+\lambda t)e^{-\lambda t}\right]$$

Where t is time, λ is the error detection rate, and K is the total number of defects or total cumulative defect rate.

-The ***inflection S model*** is another S-shaped reliability growth model that describes a software failure phenomenon with a mutual dependence of detected defects.

-Specifically, the more failures we detect, the more undetected failures become detectable.

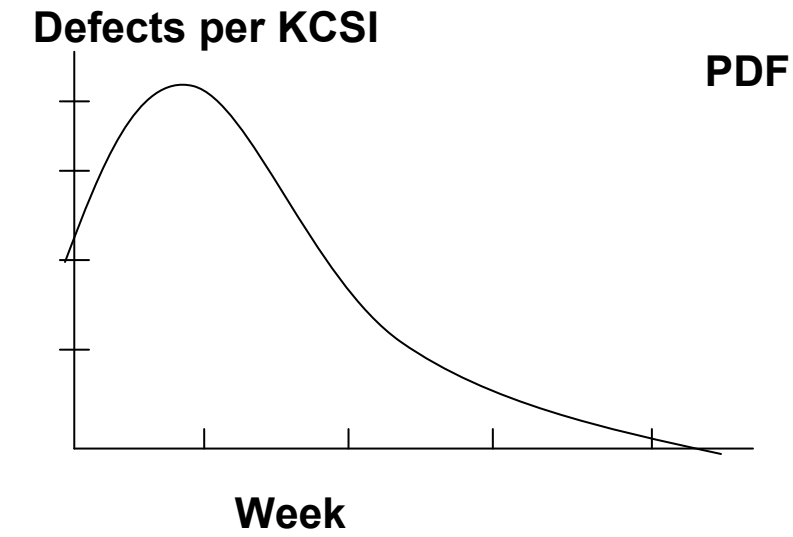
-This is a departure from previous reliability models, which make the assumption that faults occur independently.

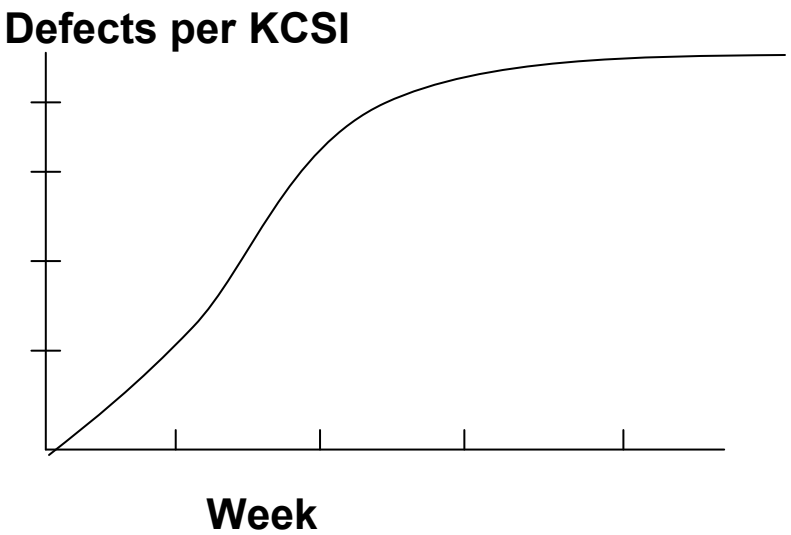
-The model is also based on the nonhomogeneous Poisson process, with a mean value function:

$$\mu(t) = K \frac{1 - e^{-\lambda t}}{1 + i e^{-\lambda t}}$$

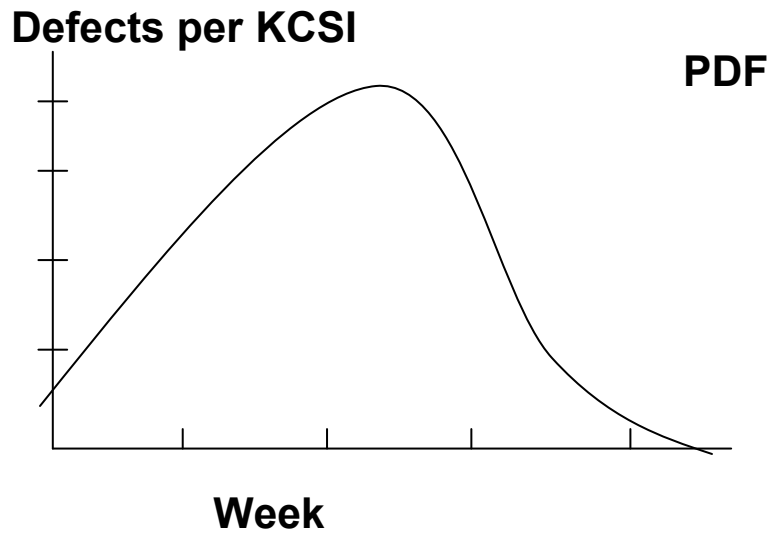
Where t is time, λ is the error detection rate, i is the inflection factor and K is the total number of defects or total cumulative defect rate.

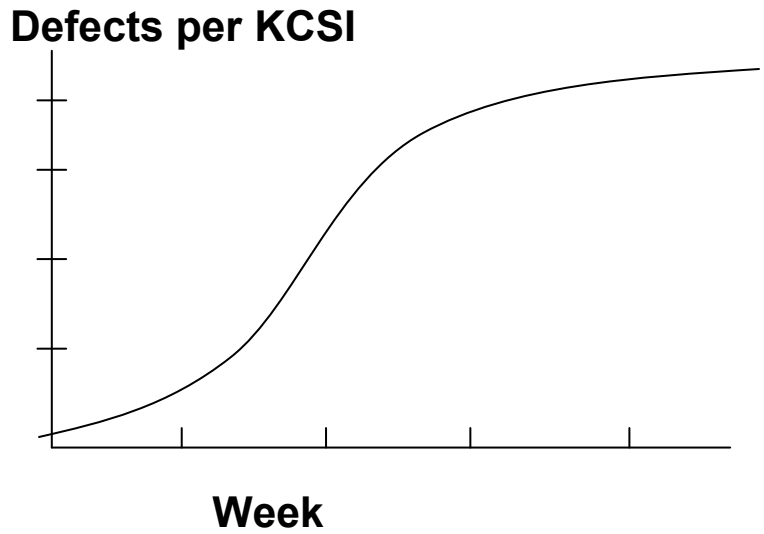
Delayed S and Inflection S Models





Delayed S model





Inflection S model

4. Modeling Process

-To model software reliability, the following steps should be followed:

1. **Examine the data:** plot the data points against time in the form of a scatter diagram, analyze the data informally and gain an insight into the nature of the process being modeled.

2. **Select a model** or several model to fit the data based on an understanding of the test process, the data, and the assumptions of the models.

3. **Estimate the parameters of the model** using statistical techniques such as maximum Likelihood or least-squares methods.

4. **Obtain the fitted model** by substituting the estimates of the parameters into the chosen model.

5. **Perform a goodness-of-fit test** and assess the reasonableness of the model. If the model does not fit, a more reasonable model should be selected.

6. **Make reliability predictions based on the fitted model.**

Example7.8.2: Modeling based on sample defect data

Week	Defects Arrival/KLOC	Defects/KLOC cumulative
1	.353	.353
2	.436	.789
3	.415	1.204
4	.351	1.555
5	.380	1.935
6	.366	2.301
7	.308	2.609
8	.254	2.863
9	.192	3.055
10	.219	3.274
11	.202	3.476
12	.180	3.656
13	.182	3.838
14	.110	3.948
15	.155	4.103
16	.145	4.248
17	.221	4.469
18	.095	4.564
19	.140	4.704
20	.126	4.830

Kolmogorov-Smirnov goodness-of-fit test for sample data

Week	Observed Defects/KLOC cumulative (A)	Model Defects/KLOC cumulative (B)	F*(x)	F(x)	|F*(x)-F(x)|
1	353	.437	0.07314	.09050	.01736
2	.789	.845	.16339	.17479	.01140
3	1.204	1.224	.24936	.25338	.00392
4	1.555	1.577	.32207	.32638	.00438
5	1.935	1.906	.40076	.39446	.00630
6	2.301	2.213	.47647	.45786	.01861
7	2.609	2.498	.54020	.51691	.2329
8	2.863	2.764	.59281	.57190	.02091
9	3.055	3.011	.63259	.62311	.00948
10	3.274	3.242	.67793	.67080	.00713
11	3.476	3.456	.71984	.71522	.00462
12	3.656	3.656	.75706	.75658	.00048
13	3.838	3.842	.79470	.79510	.00040
14	3.948	4.016	.81737	.83098	.01361
15	4.103	4.177	.84944	.86438	.01494
16	4.248	4.327	.87938	.89550	.01612
17	4.469	4.467	.92515	.92448	.00067
18	4.564	4.598	.94482	.95146	.00664
19	4.704	4.719	.97391	.97659	.00268
20	4.830	4.832	1.0000	1.0000	.00000

5. Identifying Trends in Failure Data

Overview

-Before applying any model to a set of failure data, it is advisable to determine whether the failure data does, in fact, exhibit reliability growth.

-If a set of failure data does not exhibit increasing reliability as testing progresses, there is no point in attempting to estimate and forecast the system's reliability.

-In this course, we will study two types of trend tests (supported by the CASRE tool) that can be applied to both time between failures data and failure counts data. These are the ***running arithmetic average*** and the ***Laplace test***.

Running Arithmetic Average

-The running arithmetic average is one of the simplest trend tests that can be applied to determine whether a set of failure data exhibits reliability growth.

-This test may be applied to both time between failures data and failure counts data. *For failure counts data, the test may only be applied to data in which the test intervals are of equal length.*

Running Arithmetic Average for Time between failures

-For time between failures data, the running arithmetic average after the ***i^th^*** failure has been observed, ***r(i)***, is given by:

$$r(i) = \frac{\sum_{j=1}^{i} \theta_j}{i}$$

- Where θ_j is the observed time between the ***(j-1)^st^*** and the ***j^th^*** failure**.**

-Running arithmetic average test results for time between failures data are interpreted as follows:

- If the running arithmetic average increases as the failure number increases, the time between failures is increasing. Hence the system's reliability is increasing. Reliability models may be applied if the running arithmetic average is increasing.

- Conversely, if the running arithmetic average decreases with increasing failure number, the average time between failures is decreasing, meaning that the system is becoming less reliable as testing progresses.

Example: Running Arithmetic Average - Time Between Failures Data

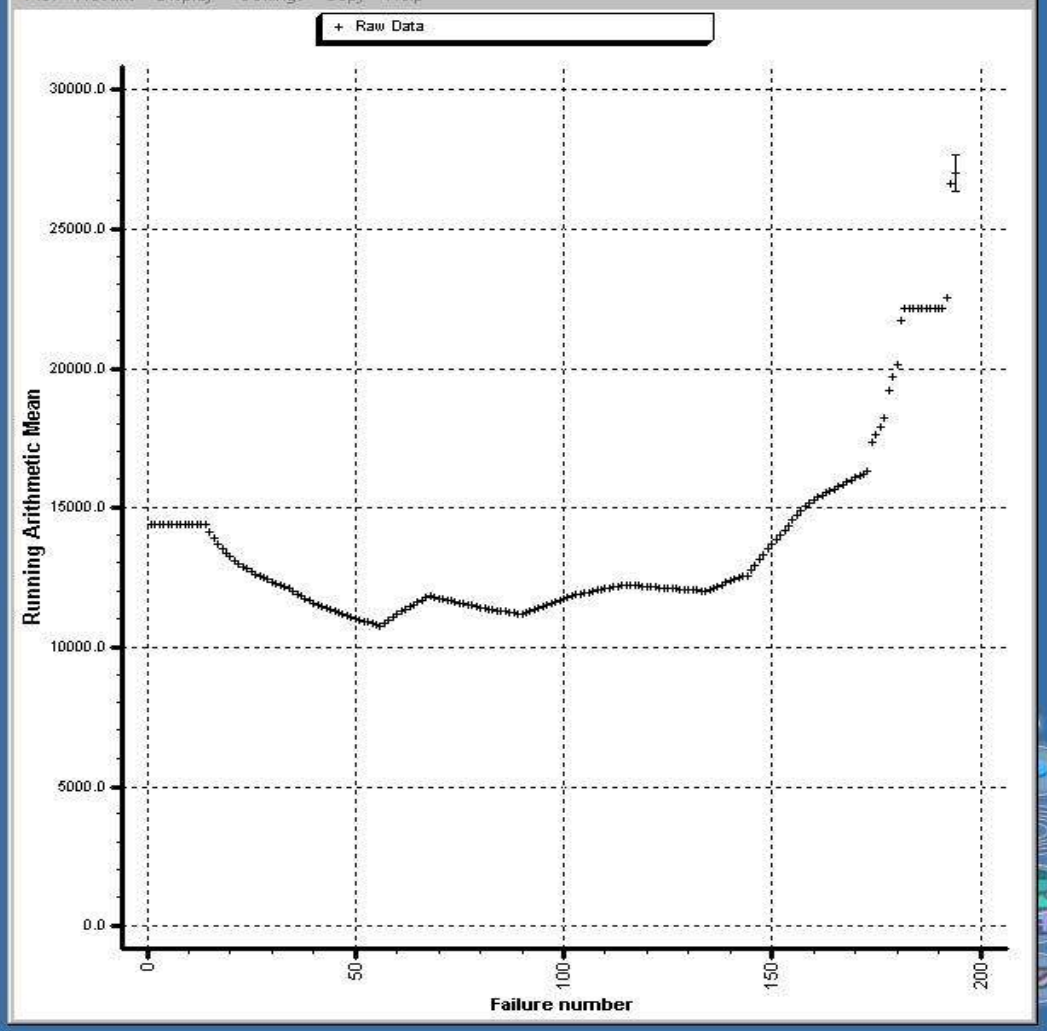
CASRE

File Edit Filters Trend Model Setup Plot Help

Error No.	Seconds Since Last Failure	Arithmetic Mean
1	1.440000e+004	1.440000e+004
2	1.440000e+004	1.440000e+004
3	1.440000e+004	1.440000e+004
4	1.440000e+004	1.440000e+004
5	1.440000e+004	1.440000e+004
6	1.440000e+004	1.440000e+004
7	1.440000e+004	1.440000e+004
8	1.440000e+004	1.440000e+004
9	1.440000e+004	1.440000e+004
10	1.440000e+004	1.440000e+004
11	1.440000e+004	1.440000e+004
12	1.440000e+004	1.440000e+004
13	1.440000e+004	1.440000e+004
14	1.440000e+004	1.440000e+004
15	1.060920e+004	1.414728e+004
16	1.060920e+004	1.392615e+004
17	1.060920e+004	1.373104e+004
18	1.060920e+004	1.355760e+004
19	1.060920e+004	1.340242e+004
20	1.060920e+004	1.326276e+004
21	1.060920e+004	1.313640e+004
22	1.060920e+004	1.302153e+004
23	1.060920e+004	1.291664e+004
24	1.060920e+004	1.282050e+004
25	1.060920e+004	1.273205e+004
26	1.060920e+004	1.265040e+004
27	1.060920e+004	1.257480e+004
28	1.060920e+004	1.250460e+004
29	1.060920e+004	1.243924e+004
30	1.060920e+004	1.237824e+004
31	1.060920e+004	1.232117e+004
32	1.060920e+004	1.226767e+004
33	1.060920e+004	1.221741e+004
34	8.765280e+003	1.211588e+004
35	8.765280e+003	1.202015e+004
36	8.765280e+003	1.192974e+004
37	8.765280e+003	1.184421e+004
38	8.765280e+003	1.176319e+004

Running Arithmetic Mean: tbe_tst2.dat

Plot Results Display Settings Copy Help



Start | 3 Window... | 2 Microsof... | Login - Micr... | 2 Adobe ... | casre-short-... | 2 NTVD... | Figure1 - Pa... | 1:39 PM

Running Arithmetic Average for Failure Counts

-For failure counts data, the running arithmetic average after the ***i^th^*** test interval has been completed, ***r(i)***, is given by:

$$r(i) = \frac{\sum_{j=1}^{i} n_j}{i}$$

- Where n_j is the number of failures that have been observed in the ***j^th^*** test interval.

-Running arithmetic average test results for failure counts data are interpreted as follows:

- If the running arithmetic average decreases as the test interval number increases, the number of failures observed per test interval is decreasing. Hence the system's reliability is increasing. For failure counts data, reliability models may be applied if the running arithmetic average is decreasing with increasing test interval number.

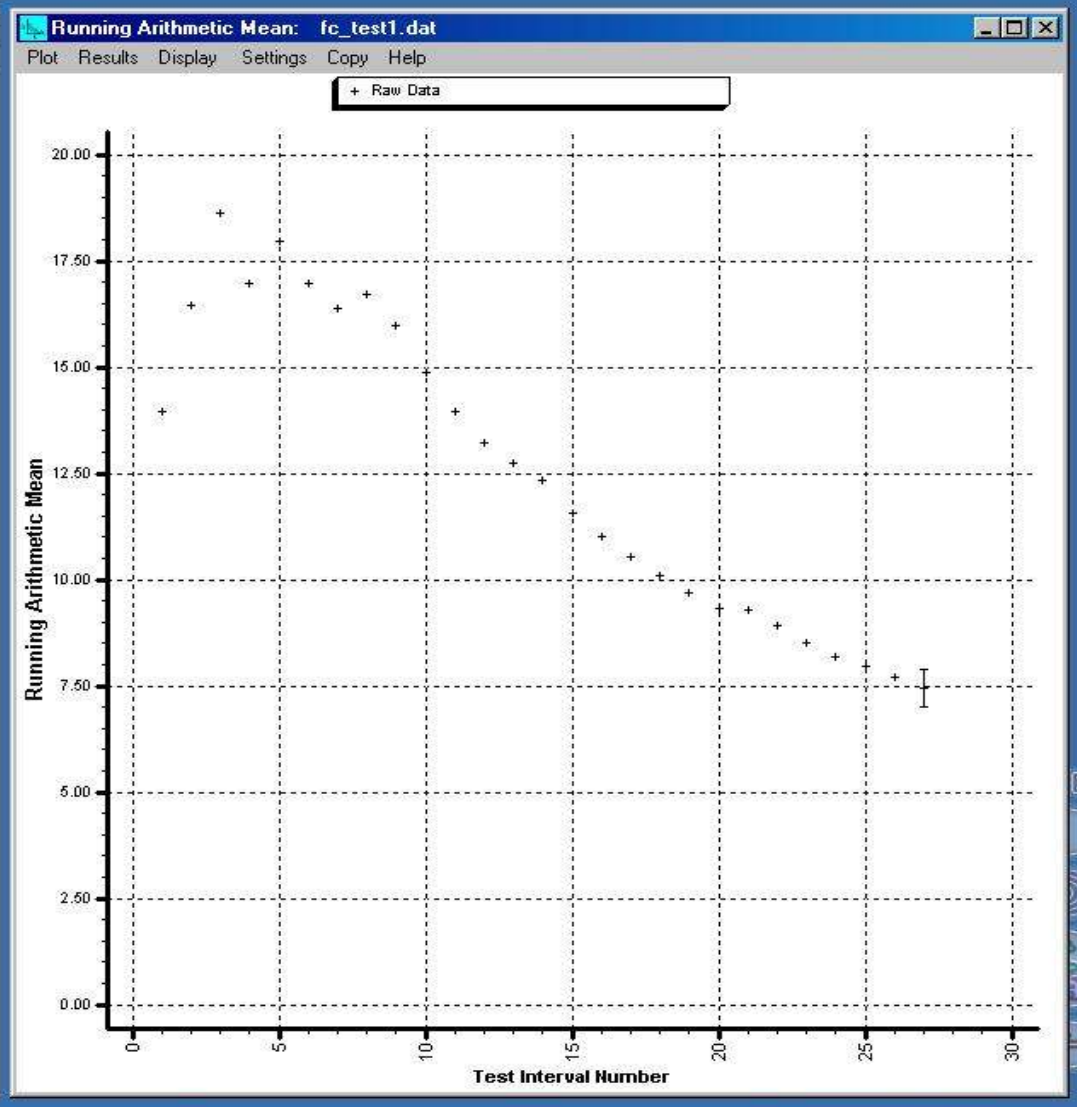
- Conversely, if the running arithmetic average increases with increasing test interval number, the average number of failures observed per test interval is increasing, meaning that the system is becoming less reliable as testing progresses.

Example: Running Arithmetic Average - Failure Counts Data

CASRE

File Edit Filters Trend Model Setup Plot Help

Test Intvl	Number of Failures	Arithmetic Mean
1	1.400000e+001	1.400000e+001
2	1.900000e+001	1.650000e+001
3	2.300000e+001	1.866667e+001
4	1.200000e+001	1.700000e+001
5	2.200000e+001	1.800000e+001
6	1.200000e+001	1.700000e+001
7	1.300000e+001	1.642857e+001
8	1.900000e+001	1.675000e+001
9	1.000000e+001	1.600000e+001
10	5.000000e+000	1.490000e+001
11	5.000000e+000	1.400000e+001
12	5.000000e+000	1.325000e+001
13	7.000000e+000	1.276923e+001
14	7.000000e+000	1.235714e+001
15	1.000000e+000	1.160000e+001
16	3.000000e+000	1.106250e+001
17	3.000000e+000	1.058823e+001
18	2.000000e+000	1.011111e+001
19	3.000000e+000	9.736842e+000
20	2.000000e+000	9.350000e+000
21	9.000000e+000	9.333333e+000
22	1.000000e+000	8.954545e+000
23	0.000000e+000	8.565217e+000
24	0.000000e+000	8.208333e+000
25	3.000000e+000	8.000000e+000
26	1.000000e+000	7.730769e+000
27	1.000000e+000	7.481482e+000



Start | 3 Window... | 2 Microsof... | Login - Micr... | casre-short-... | Figure9 - Pa... | Adobe Rea... | 2 NTVD... | 2:58 PM

Appendix: Laplace Trend Test

Laplace Test

-The Laplace test may be applied to both time between failures data and failure counts data. *For failure counts data, the test may only be applied to data in which the test intervals are of equal length.*

-Laplace test results are interpreted as follows:

- Reject the null hypothesis that occurrences of failures follow a Homogeneous Poisson Process (a Poisson process in which the rate remains unchanged over time) in favor of the hypothesis of reliability growth at the α% significance level if the test statistic is less than or equal to the value at which the cumulative distribution function for the normal distribution is $\alpha/100$.

  -For example, if α is set to 5%, the value of the cumulative normal distribution function is approximately –2. If the value of the Laplace test were to be –2 for a set of failure data, then we could reject the null hypothesis of occurrences of failures following a Homogeneous Poisson Process at the 5% significance level.

- Reject the null hypothesis that occurrences of failures follow a Homogeneous Poisson Process in favor of the hypothesis of reliability decrease at the α% significance level if the test statistic is greater than or equal to the value at which the Cumulative Distribution Function (CDF) for the normal distribution is $(1-\alpha)/100$.

- Reject the null hypothesis that there is either reliability growth or reliability decrease in favor of the hypothesis that there is no trend at the α% significance level if the statistic is between the values at which the CDF for the normal distribution is $\alpha/200$ and $(1-\alpha/2)/100$. If α = 5%, the Laplace test statistic must lie between –2 and 2 for this hypothesis to be accepted.

-For time between failures data, the value of the Laplace test statistic after the ith failure has been observed, ***u(i)***, is given by the following equation,

$$u(i) = \frac{\frac{1}{i-1}\sum_{n=1}^{i-1}\sum_{j=1}^{n}\theta_j - \frac{\sum_{j=1}^{i}\theta_j}{2}}{\sum_{j=1}^{i}\theta_j\sqrt{\frac{1}{12(i-1)}}}$$

- Where θ_j is the elapsed testing time between the jth and $(j-1)^{st}$ failures.

-For failure counts data, the Laplace test statistic is given by:

$$u(i) = \frac{\sum_{i=1}^{k}(i-1)n(i) - \frac{(k-1)}{2}\sum_{i=1}^{k}n(i)}{\sqrt{\frac{k^2-1}{12}\sum_{i=1}^{k}n(i)}}$$

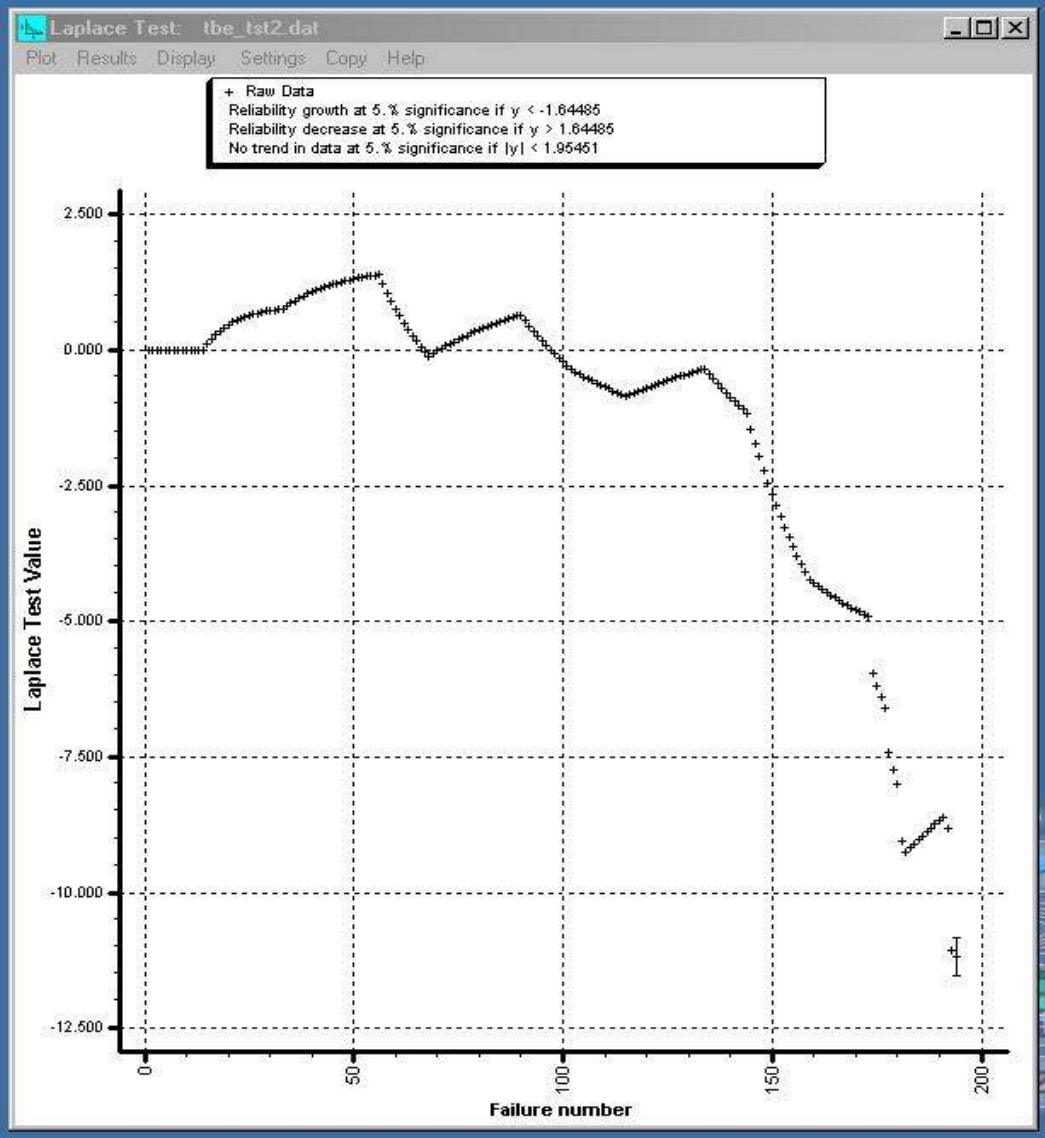
- Where:
  - The entire testing period, represented by the interval ***[0,T]*** has been divided into ***k*** equal length time units.
  - The number of failures observed during the ith time unit is ***n(i)***.

Example: Laplace Test Applied to Time Between Failures data

CASRE

File Edit Filters Trend Model Setup Plot Help

Error No.	Seconds Since Last Failure	Laplace Test
1	1.440000e+004	0.000000e+000
2	1.440000e+004	0.000000e+000
3	1.440000e+004	0.000000e+000
4	1.440000e+004	0.000000e+000
5	1.440000e+004	0.000000e+000
6	1.440000e+004	0.000000e+000
7	1.440000e+004	0.000000e+000
8	1.440000e+004	0.000000e+000
9	1.440000e+004	0.000000e+000
10	1.440000e+004	0.000000e+000
11	1.440000e+004	0.000000e+000
12	1.440000e+004	0.000000e+000
13	1.440000e+004	0.000000e+000
14	1.440000e+004	0.000000e+000
15	1.060920e+004	1.157686e-001
16	1.060920e+004	2.130359e-001
17	1.060920e+004	2.953441e-001
18	1.060920e+004	3.654260e-001
19	1.060920e+004	4.254181e-001
20	1.060920e+004	4.770123e-001
21	1.060920e+004	5.215631e-001
22	1.060920e+004	5.601647e-001
23	1.060920e+004	5.937091e-001
24	1.060920e+004	6.229311e-001
25	1.060920e+004	6.484390e-001
26	1.060920e+004	6.707396e-001
27	1.060920e+004	6.902596e-001
28	1.060920e+004	7.073576e-001
29	1.060920e+004	7.223387e-001
30	1.060920e+004	7.354642e-001
31	1.060920e+004	7.469563e-001
32	1.060920e+004	7.570079e-001
33	1.060920e+004	7.657840e-001
34	8.765280e+003	8.214279e-001
35	8.765280e+003	8.728323e-001
36	8.765280e+003	9.203743e-001
37	8.765280e+003	9.643912e-001
38	8.765280e+003	1.005186e+000



Start | 3 Window... | 2 Microsof... | Login - Micr... | 2 Adobe ... | casre-short-... | 2 NTVD... | Figure2 - Pa... | 1:41 PM

Example: Laplace Test Applied to Failure Counts Data

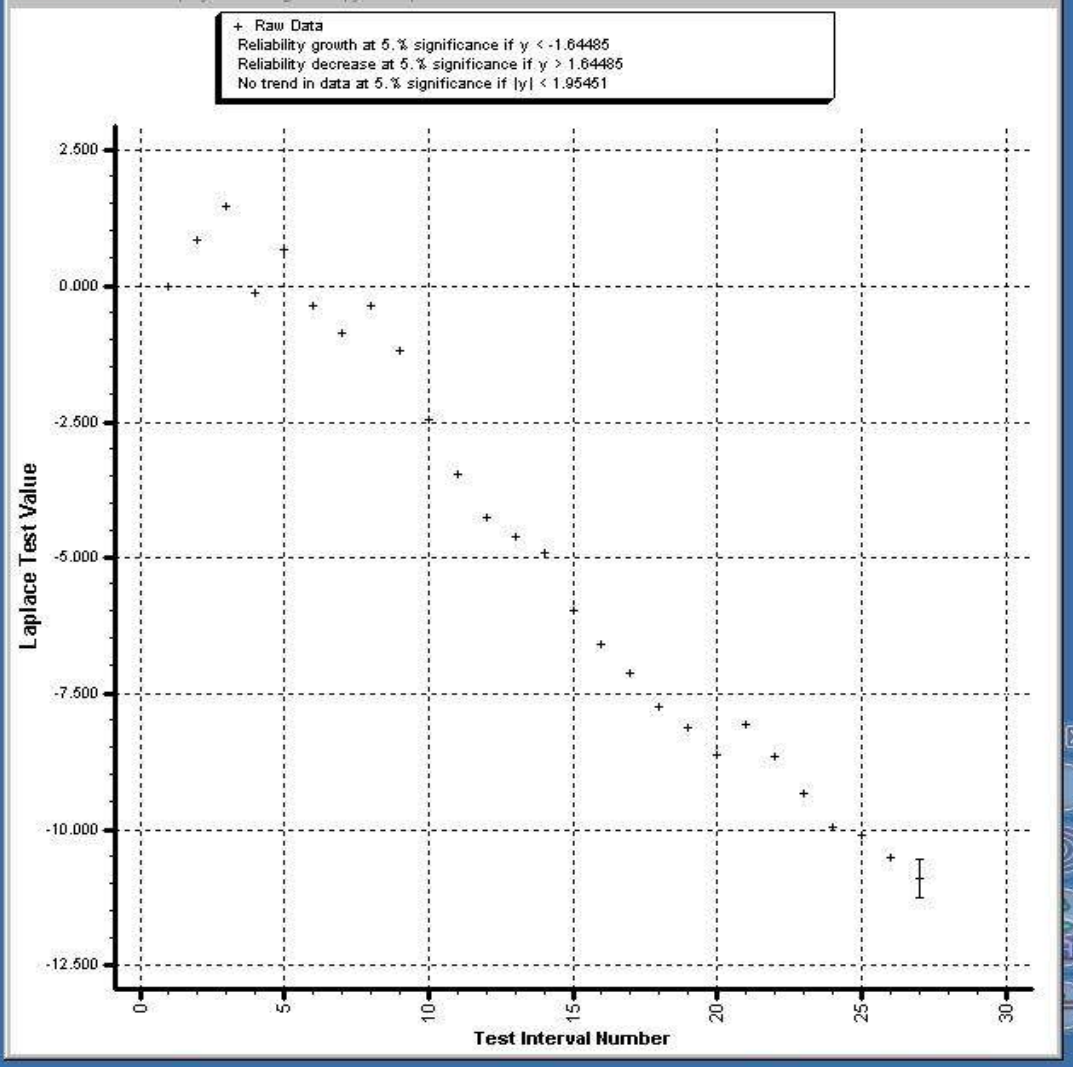
CASRE

File Edit Filters Trend Model Setup Plot Help

Test Intvl	Number of Failures	Laplace Test
1	1.400000e+001	0.000000e+000
2	1.900000e+001	8.703883e-001
3	2.300000e+001	1.472971e+000
4	1.200000e+001	-1.084652e-001
5	2.200000e+001	6.708204e-001
6	1.200000e+001	-3.478626e-001
7	1.300000e+001	-8.392543e-001
8	1.900000e+001	-3.393208e-001
9	1.000000e+001	-1.161895e+000
10	5.000000e+000	-2.438631e+000
11	5.000000e+000	-3.440119e+000
12	5.000000e+000	-4.238580e+000
13	7.000000e+000	-4.605056e+000
14	7.000000e+000	-4.894267e+000
15	1.000000e+000	-5.948284e+000
16	3.000000e+000	-6.579266e+000
17	3.000000e+000	-7.120393e+000
18	2.000000e+000	-7.729492e+000
19	3.000000e+000	-8.120988e+000
20	2.000000e+000	-8.604650e+000
21	9.000000e+000	-8.044900e+000
22	1.000000e+000	-8.641568e+000
23	0.000000e+000	-9.323098e+000
24	0.000000e+000	-9.947763e+000
25	3.000000e+000	-1.009018e+001
26	1.000000e+000	-1.050025e+001
27	1.000000e+000	-1.087616e+001

Laplace Test: fc_test1.dat

Plot Results Display Settings Copy Help



Start | 3 Window... | 2 Microsof... | Login - Micr... | casre-short-... | Figure3 - Pa... | Adobe Rea... | 2 NTVD... | 2:59 PM