2

RL-TR-93-87
Final Technical Report
May 1993

USING MARKOV CHAINS TO MODEL THE ERROR BEHAVIOR OF DATA COMMNICATIONS CHANNELS

Mississippi State University

Dr. Wayne D. Smith, Karen D. Burns, Jane N. Moorhead

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

Rome Laboratory
Air Force Materiel Command
Griffiss Air Force Base, New York

RL-TR-93-87 has been reviewed and is approved for publication.

APPROVED: Jeffrey K. Long

JEFFREY K. LONG, Capt, USAF
Project Engineer

FOR THE COMMANDER John A. Graniero

JOHN A. GRANIERO
Chief Scientist for C3

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE May 1993	3. REPORT TYPE AND DATES COVERED Final Mar 92 - Jan 93

4. TITLE AND SUBTITLE
USING MARKOV CHAINS TO MODEL THE ERROR BEHAVIOR OF DATA COMMUNICATIONS CHANNELS

5. FUNDING NUMBERS
C - F30602-92-C-0023
PE - 33126F
PR - 2022
TA - 01
WU - P5

6. AUTHOR(S)
Dr. Wayne D. Smith, Karen D. Burns, Jane N. Moorhead

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Mississippi State University
Department of Computer Science
Mississippi State, MS 39762

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Rome Laboratory (C3DA)
525 Brooks Rd
Griffiss AFB NY 13441-4505

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
RL-TR-93-87

11. SUPPLEMENTARY NOTES
Rome Laboratory Project Engineer: Jeffrey K. Long, Capt, USAF/C3DA/(315) 330-7751

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (Maximum 200 words)

This report is intended to present a description of the work performed for a project to model data communications error patterns using Markov Models. It details a method for arriving at a final Markov model starting from a collection of channel error statistics, presents a rough Markov Model for a Meteor Burst Channel and lastly presents a theoretical model for a channel that presents a non-specific, "N-Bits in error over M-Percent" channel error pattern.

14. SUBJECT TERMS
Bit-Error-Rate Models, Markov Models, Data Communications

15. NUMBER OF PAGES
64

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	18. SECURITY CLASSIFICATION OF THIS PAGE UNCLASSIFIED	19. SECURITY CLASSIFICATION OF ABSTRACT UNCLASSIFIED	20. LIMITATION OF ABSTRACT UL

Table of Contents

1.0 Foreword

This report is intended to present a description of the work performed for the project to model data communications error through the use of Markov models. It is intended to provide a more detailed narrative description of the work performed than is usually included in the quarterly progress reports, but eliminates the accounting information that is available in the quarterly reports.

This document, being the final project report, covers the entire time period of the project. It includes all work accomplished for the duration of this project and supercedes any and all previous project narratives.

In order to make this document more understandable, the sections have been numbered in accordance with the Statement of Work associated with this project. Hence, most of the material is covered under section 4 and the numbers do not necessarily fall into sequence. Most of the figures mentioned in the text are included in Appendix A of this document.

2.0 Strategy for Reducing Gap Length Distribution to a Markov Model

The simulation of errors occurring over a communications channel provides a less costly alternative for testing communications systems than running the system over the actual channel. A simple method of simulating these errors assumes that errors are independent and produces the correct percentage of errors for a given channel. For actual channels, errors are not independent, but tend to occur in "bursts."

Such "bursty" channels, or channels "with memory", have been successfully modelled using finite state Markov chains. After some introductory information about Markov models, a procedure for obtaining Markov models for communications channels will be presented.

2.1 Markov Processes

Markov processes are a subset of stochastic processes. A stochastic process is a family of random variables defined over some sample space. The elements of the sample space can represent different states. A process changes states as time progresses. Discrete-time processes can change states only at the end of a time interval of some given length. Stationary processes are time independent. They exhibit the same behavior regardless of when they are observed.

A Markov process is a discrete-time stochastic process with a finite state space that satisfies the Markov property. Isaacson and Madsen define the Markov property by stating, "A stochastic process $\{X_k\}$, k=1, 2, 3, . . . with state space $S = \{1, 2, 3, \ldots\}$ is said to satisfy the Markov property if for every n and all states $i_1, i_2, \ldots, i_n$ it is true that

$$P(X_n=i_n|X_{n-1}=i_{n-1},X_{n-2}=i_{n-2}, \ldots ,X_1=i_1) = P(X_n=i_n|X_{n-1}=i_{n-1}).''$$

In other words, the current state of a Markov process is conditional only upon the previous state and not upon any other past states[12].

A Markov model is similar to a finite state machine. The Markov process transitions from state to state according to transition rules. However, instead of having transition rules based on symbols read from some input, Markov processes have transition rules based on random probability. Markov models can be represented as NxN matrices with entries $p_{i,j}$. For such a matrix, N is the number of states, i and j range from 1 to N, and $p_{i,j}$ is the probability of transitioning from some state i to some state j with no intermediate states.

A Markov process travels at random between the possible states. The probability of run lengths corresponding to each state can be expressed as weighted exponentials. This observation is based on joint probabilities. For example, let $P_1(x)$ be the probability that a process is in state 1 for at least x time intervals and assume the Markov model in question has only 2 states. Then $P_1(x)$ is equal to the joint probability of the process transitioning from state 2 to state 1 followed by the process transitioning from state 1 back to itself at least x-1 times. This is mathematically represented as

$$P_1(x) = p_{21}p_{11}^{x-1}$$

or

$$P_1(x)=Ae^{ax}$$
$$A=\frac{p_{21}}{p_{11}}$$
$$a=\ln(p_{11})$$

This expression could easily be expanded for models with more than two states. If the Markov model had three states, the following expression would result:

$$P_1(x)=(p_{21}+p_{31})p_{11}^{x-1}.$$

2.2 Modelling Communications Channels as Markov Processes

The errors produced by a communications channel can be represented as a string of bits. In this bit string, a bit has the value 1 if the channel transmits an incorrect bit and 0 if the channel transmits the correct bit. When this representation is used, the transmitted bit string XOR-ed with the error bit string produces the received bit string [8].

One statistic that can be analyzed for a given error bit string is the error-free run distribution, or gap length distribution. A gap is defined as a series of consecutive error-free bits bounded on each end by at least one error bit. A gap begins with the first 0 following an error bit and ends with the last 0 preceding the next error bit. The gap length distribution is the probability that a gap of length m or greater will occur, which can be written as $P(0^m|1)$ for m = 1, 2, 3. . . .

Another distribution of interest is the burst length distribution. A burst can be defined as a series of consecutive error bits bounded on each end by at least one non-error bit. A burst begins with the first 1 following a non-error bit and ends with the last 1 preceding the next non-error bit. Similar to the gap length distribution, the burst length distribution is the probability that a burst of length m or greater will occur, or $P(1^m | 0)$ for $m = 1, 2, 3, \ldots$

Stephen Tsai [4] proposes an N-state Markov model to simulate errors that occur on binary communications channels with memory. Each state in the model has an associated output of eitner 0 or 1. As the corresponding process transitions from state to state, an error bit string is produced. The model proposed by Tsai has N-1 non-error states and a single error producing state. For ease of notation, the error state is always the N^{th} state.

Tsai also specifies that transitions between two non-error states will not occur. When the process is in a non-error state, the only transitions that can occur are the transition to the error-producing state or the transition from the current state back to itself. The corresponding probability matrix would have entries along the diagonal for transitions from a state to itself, along the N^{th} column for transitions from non-error states to the error state, and along the N^{th} row for transitions from the error state to the non-error states.

For a Markov model with these criteria, the gap length distribution corresponds to the probability of runs of given length in the non-error states. For a given non-error state, i, the probability of its producing a run of x non-error bits is

$$P_i(x) = A_i e^{a_i x}$$
$$A_i = \frac{p_{Ni}}{p_{ii}}$$
$$a_i = \ln(p_{ii})$$

For an N-state Markov model where each non-error state contributes an exponential term, the probability of a gap of length x occurring is

$$P(0^{x}|1) = A_1 e^{a_1 x} + A_2 e^{a_2 x} + \ldots + A_{N-1} e^{a_{N-1} x}$$

The burst length distribution corresponds to the probability of runs of given length in the error state. The probability of a burst of length x occurring is

$$\begin{aligned} P(1^{x}|0) &= B e^{bx} \\ B &= \frac{(p_{1,N} + p_{2,N} + \ldots + p_{N-1,N})}{p_{NN}} \\ b &= \ln(p_{NN}). \end{aligned}$$

Tsai's proposed Markov model for communications channels with memory is based on the observation that the gap length distribution for a channel can be closely approximated by the sum of weighted exponential terms as described above. Several researchers have shown that this is possible for actual channel data. This model also assumes that the burst length distribution can be expressed as a single weighted exponential.

2.3 Procedure for Transformi‥ Channel Data to Markov Models

To obtain a Marko‥ ‥odel for a given binary communications channel, gap length data must be collected. A count of the number of gaps of each length is necessary. From these cou‥ts, observed values for $P(0^m | 1)$ can be computed. Once these values are computed, a curve with the form mentioned above is fit to the resulting points.

The method for fitting a curve to data points depends on the size of the sample and the level of confidence in those data points. For a small sample size where confidence in each data point is high, it may be desirable to interpolate a curve that includes all points. For larger samples where some variance of the data points is known to exist, finding a curve that comes close to all the points but doesn't necessarily include each one is more appropriate.

Computer tools exist that make curve fitting easier. However, these tools still require some human involvement. The user must usually specify the form of the resulting equation and some initial values for the parameters of that equation. These initial guesses can often make the difference between a good final fit and a bad final fit. Even if a good final fit results, bad initial guesses cause the curve-fitting program to perform more iterations before converging to a final equation.

The following explanation addresses how to make good initial parameter guesses for an equation that takes the form of the sum of weighted exponential terms. These guesses will be made based on visual inspection of the plotted data points and will be used as input to a non-linear least-squares regression tool.

For all examples herein, the data points represent statistics obtained from simulated data communications channels. The x-value of each point represents the length in bits of a gap between bursts of errors. The y-value represents the probability that a gap longer than or equal to x bits will occur.

2.3.1 Initial Guesses for One Term

The data points can be fit to an equation that is the sum of one or more exponential terms according to the form presented above. The parameters $\mathbf{A}_i$ and $\mathbf{a}_i$ must be determined for each term to provide the closest fit to the data points involved. Initially, we will consider the simplest case of an equation with only one term, $\mathbf{A}e^{\mathbf{a}x}$.

Initial guesses can be made from a plot of the error-free run distribution data points. The exponential curve where both **A** and **a** are equal to 1 is shown in Figure 1. For the gap length data involved in this discussion, x (representing the number of bits in an error-free gap) will never have a negative value. Therefore, we can focus our attention to only positive x values.

Figures 2 and 3 show the effects of varying **A** values and **a** values. As **A** increases, the height of the curve increases. As **a** increases, the drop in y values becomes more gradual. These two observations are the basis for our initial guesses.

Figure 2 shows that **A** is related to the maximum height of the curve. More specifically, **A=y** where $\mathbf{x}\to\mathbf{0}$. However, the minimum x value possible for the data points involved is 1, and determining from those data points where the curve will cross the y-axis is often difficult. This task is simplified by plotting the data points with the y-axis

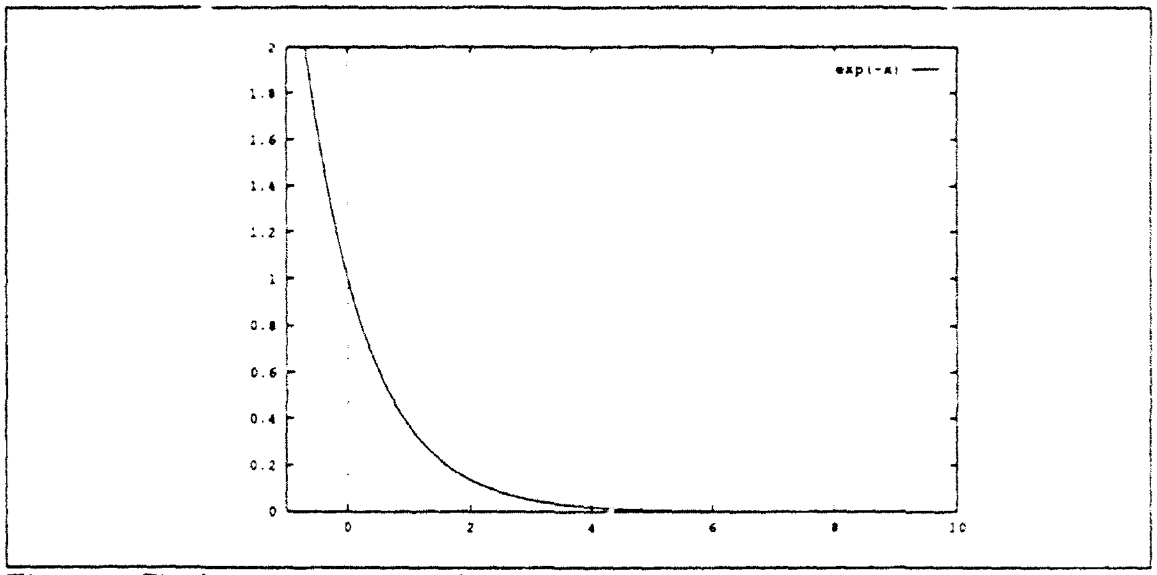

Figure 1. Single term exponential curve

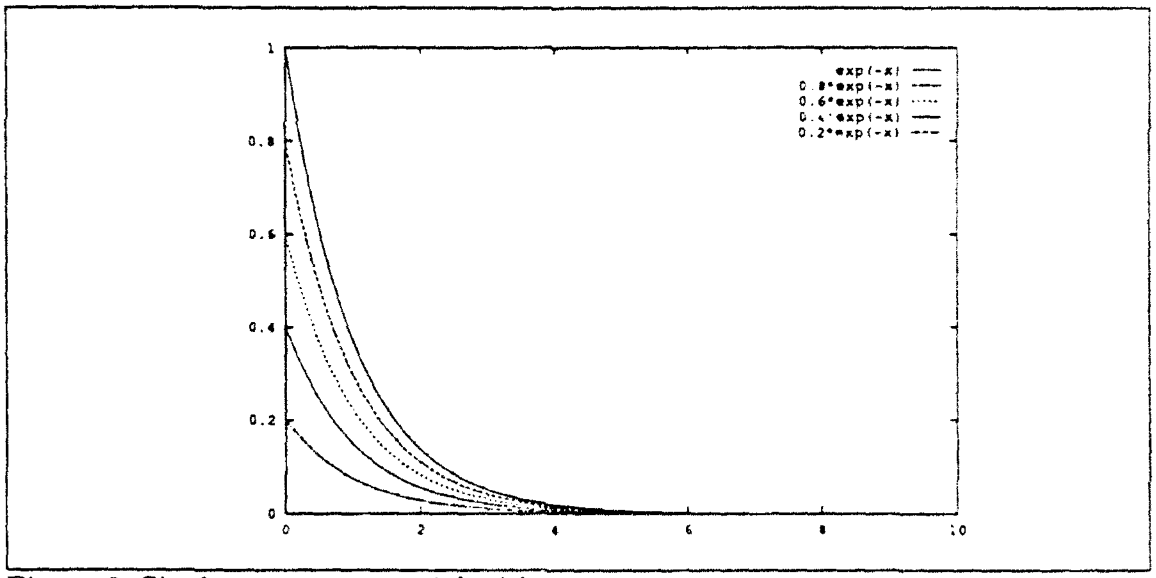

Figure 2. Single term exponential with varying A.

log scale. The curve becomes a straight line which can be extended to the y-axis, thus providing a close approximation for **A**.

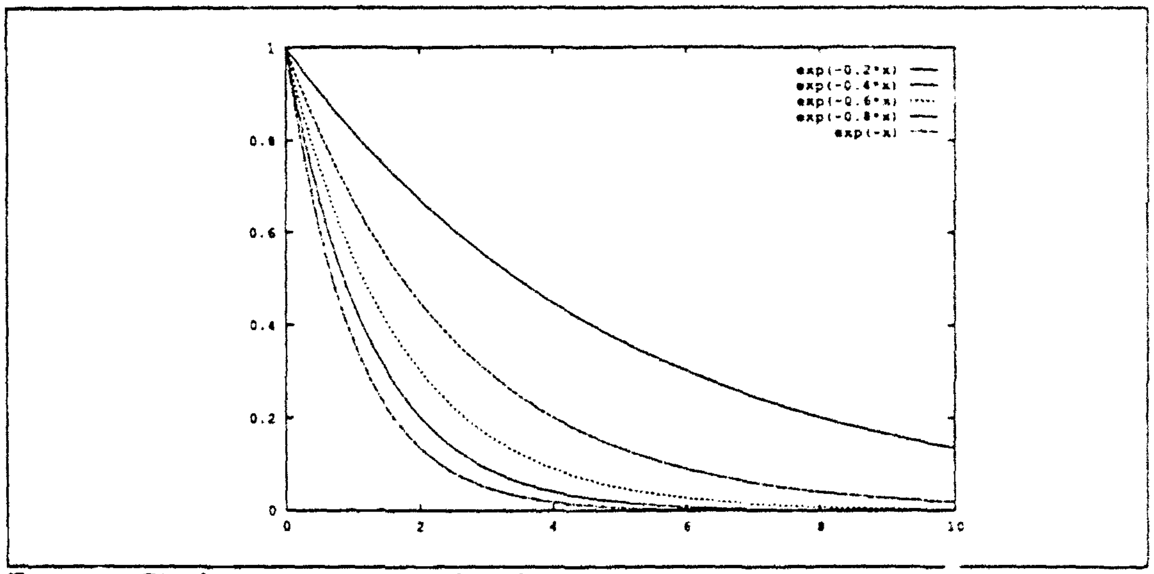

Figure 3. Single term exponential with varying a.

Figure 3 shows that **a** is related to the rate at which the y values decrease. This property can be better observed by again plotting with the y-axis log scale. Then **a=slope**. This slope can be determined by plotting the data points as **(x,*ln*(y))** rather than **(x,y)** and dividing the increase in the y direction by the increase in the x direction.

As an example of making initial parameter guesses from data points, examine Figure 4. Replotting with log scale y (Figure 5) makes determining the y-intercept easy. The value for **A** is approximately 0.9. The plot of **(x,*ln*(y))** pairs (Figure 6) produces a line that decreases 3 in the y direction over the x range 1-6. Therefore, **a** is approximately -3/5 or -0.6. (The actual equation used to simulate these data points was $0.8784e^{(-0.5407*x)}$).

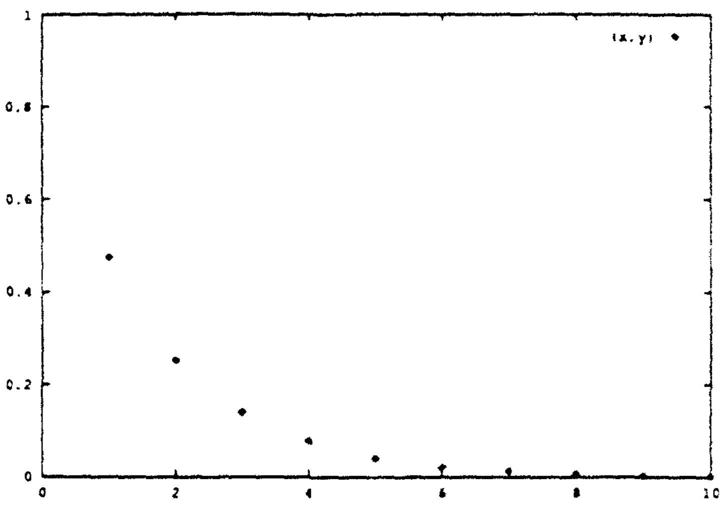

Figure 4. Error-free run data produced by Markov model simulation.

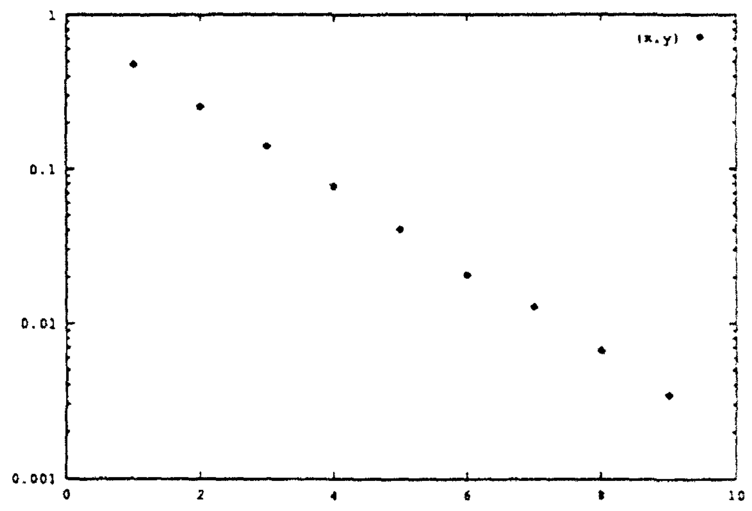

Figure 5. Error-free run data plotted with logscale y.

2.3.2 Initial Guesses for Multiple Terms

For actual binary communications channels, the gap length distribution often corresponds to an equation with more than one exponential term. Making initial guesses

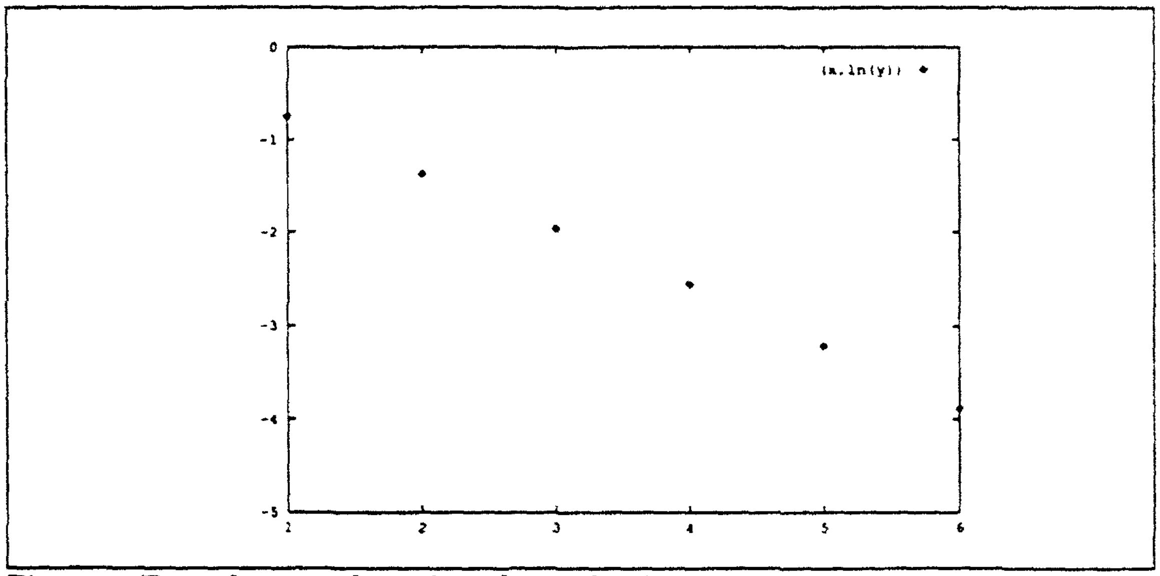

Figure 6. Error-free run data plotted as (x,ln(y)).

for such an equation is an extension of guessing for one term. As for a single term, the data is plotted with the y-axis logscale to simplify the task of making initial guesses.

When a single exponential term was plotted with the y-axis logscale, the resulting plot was a straight line. For more than one term, the plot will show linear segments separated by curved portions. The number of linear segments suggests the number of exponential terms necessary to approximate the data.

For each term, the $\mathbf{A}_i$ value is related to the y-intercept of the corresponding line segment. Because the terms are added together, the $\mathbf{A}_i$ value for each term is the y-intercept of the line segment minus the y-intercept of the previous term, starting with the term that has the smallest y-intercept.

Once the $\mathbf{A}_i$ values have been calculated, the $\mathbf{a}_i$ values are determined based on the slope of the curve within the range of each line segment. As for the single exponential

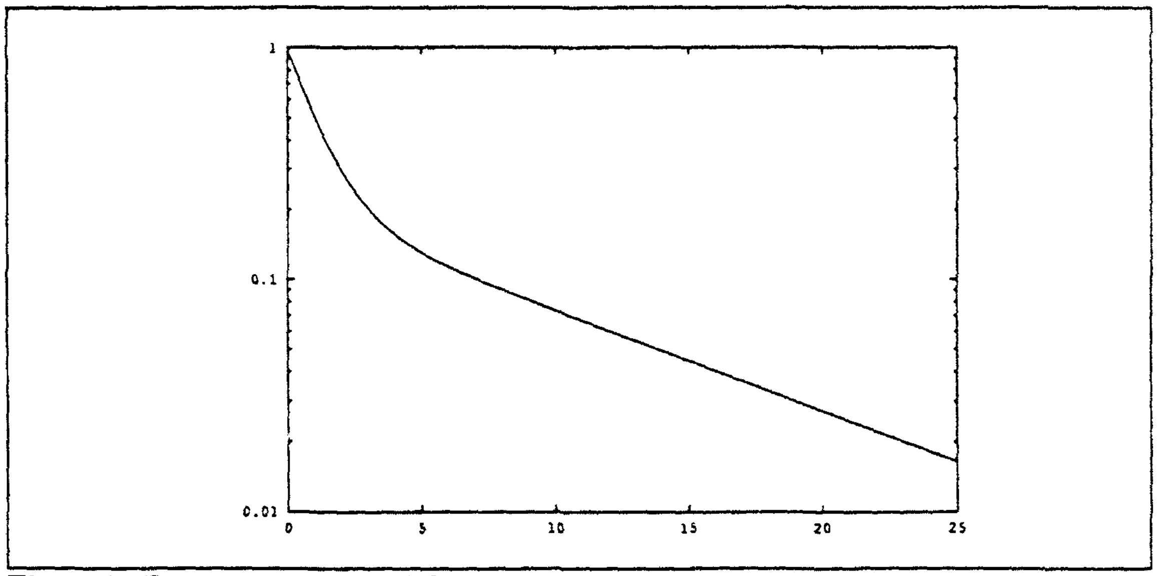

Figure 7. Two term exponential.

term, the data must be plotted as (**x**,$\boldsymbol{ln}$(**y**)) to determine the slope. Figure 7 shows the curve of an equation with two weighted exponential terms. The plot can be described as a line segment approximately between 5 and 25 along the x-axis, a line segment approximately between 0 and 3 along the x-axis, and a curved portion between the two line segments. Starting with the line segment with the lowest y-intercept, the value for $\mathbf{A}_1$ is 0.2. The value for $\mathbf{A}_2$ is 1.0-$\mathbf{A}_1$, or 0.8.

The plot of (**x**,$\boldsymbol{ln}$(**y**)) between 5 and 25 is shown in Figure 8. From this plot, the slope of this line segment can be calculated as (-4.75 - -2.75)/(25 - 5) or -0.1. This is the estimated value of $\mathbf{a}_1$. The plot of (**x**,$\boldsymbol{ln}$(**y**)) between 0 and 3 is shown in Figure 9. From this plot, the slope can be calculated as (-2.0 - 0)/(3 - 0) or -0.67. The initial guess for this equation would be $0.2e^{-0.1x} + 0.8e^{-0.67x}$. The resulting curve plotted with the original curve is shown if Figure 10.

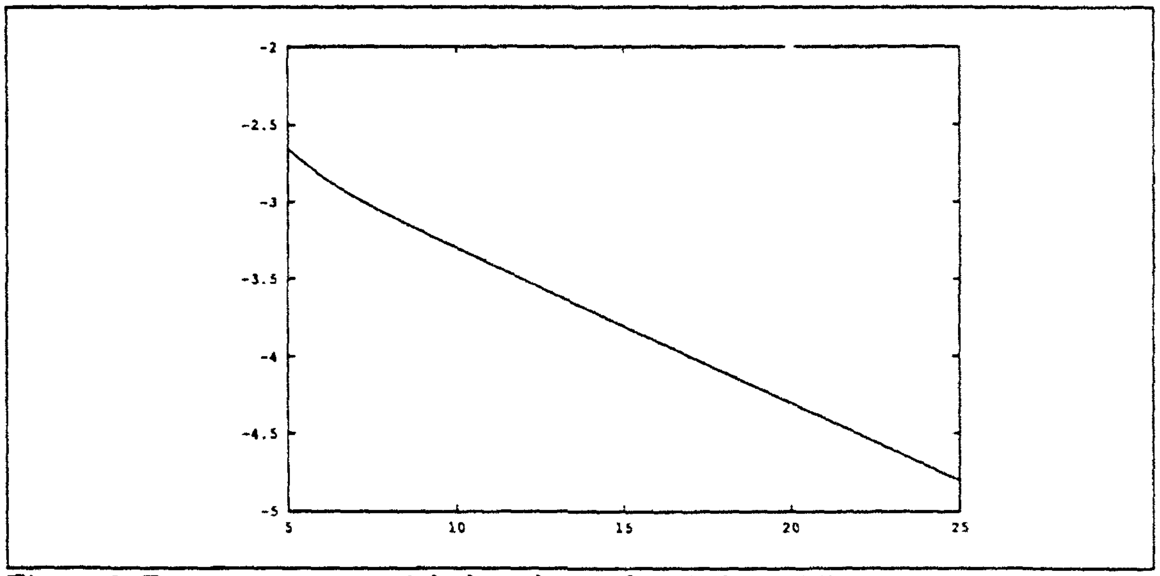

Figure 8. Two term exponential plotted as (x,ln(y)) {5<x<25}.

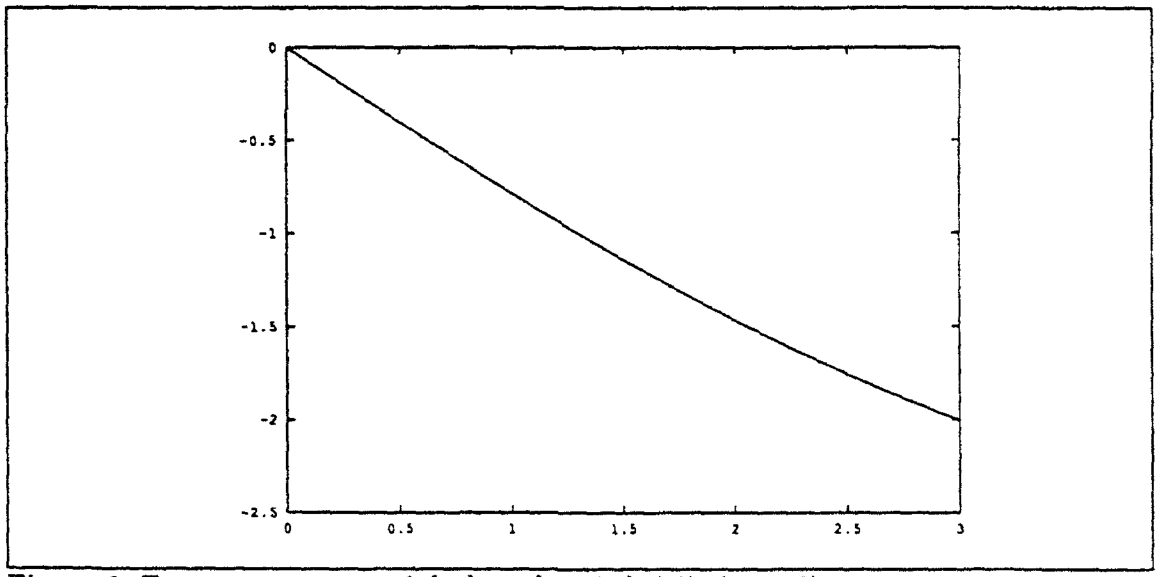

Figure 9. Two term exponential plotted as (x,ln(y)) {0<x<3}.

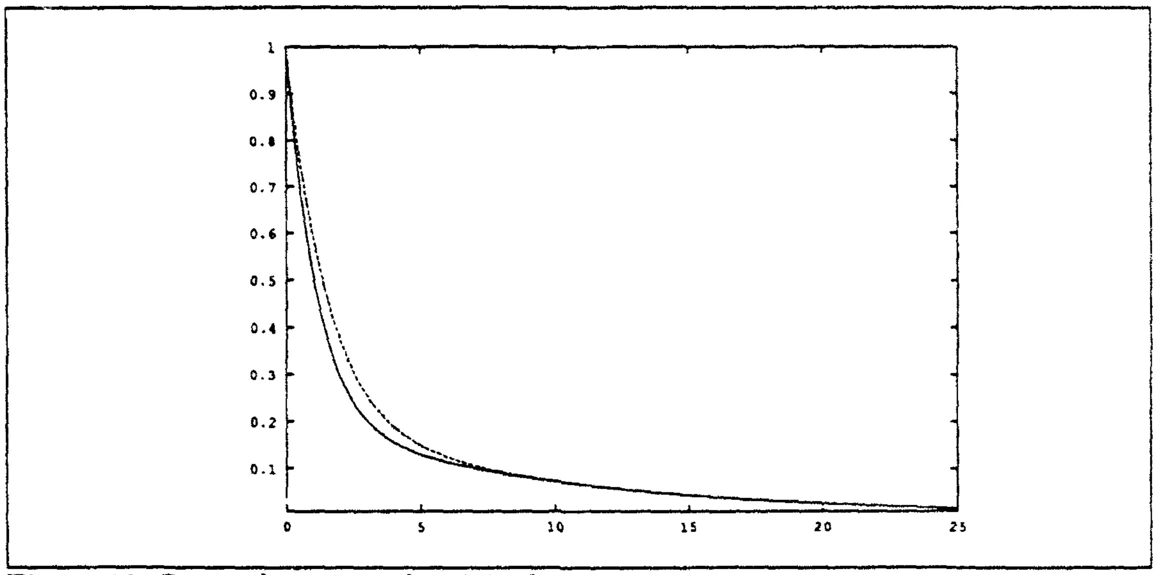

Figure 10. Guessed curve with original curve.

2.3.3 Obtaining the Markov Probability Matrix

The initial guesses obtained as described above are used as input to a software package that performs a least-squares fit to the original data points. For actual communications channels, the task of segmenting the data into line segments can be difficult. After the fitted curve is obtained, the resulting model should be analyzed to see if an additional term should be added. This can be done by plotting the fitted curve concurrently with the original data and observing whether any range of points lies far away from the fitted curve.

Once a satisfactory fitted curve has been obtained, its parameters are used to calculate the entries for a Markov probability matrix. The formulas used to compute these entries appear below. $\{1 \leq i \leq (N-1)\}$

$$p_{i,i} = e^{a_i}$$
$$p_{i,N} = 1 - p_{i,i}$$
$$p_{N,i} = A_i e^{a_i}$$
$$p_{N,N} = 1 - (p_{1,N} + p_{2,N} + \dots + p_{N-1,N})$$

3.0 Strategy for Evaluating the Markov Model

As was described in the preceding section, the use of Markov models for simulating communications channels is based on the observation that the gap length distribution for such channels can be expressed as the sum of weighted exponential terms. The resulting Markov model can be evaluated by how well the fitted exponential equation fits the original data. Three possible means of analyzing that fit are the sample standard deviation (s), the correlation coefficient (R^2), and the plot of residuals.

The sample standard deviation is the square root of the variation of the observations being analyzed.

$$s = \sqrt{\frac{\sum_{i=1}^{n} (x_1 - \bar{x})^2}{n-1}}$$

Standard deviation is preferred over variance because it can be expressed in the same units as the observations. The standard deviation approaches 0 for a perfect fit.

The correlation coefficient expresses the proportion of the variation of y's that can be attributed to the specified relationship with x.

$$r^2 = \frac{(S_{xy})^2}{S_{xx} \cdot S_{yy}}$$

where

$$S_{xx} = \sum x^2 - \frac{1}{n}(\sum x)^2$$
$$S_{yy} = \sum y^2 - \frac{1}{n}(\sum y)^2$$
$$S_{xy} = \sum xy - \frac{1}{n}(\sum x)(\sum y)$$

The correlation coefficient approaches 1 for a perfect fit.

Residuals are the values for each data point representing the distance in the y direction from the point to the fitted curve. Curve fitting is done to minimize the sum of the squared residual values. When the values for all residuals are plotted, ranges along the fitted curve that do not fit the data points are made evident by groups of residuals with relatively large values. Such plots can be used to determine whether additional terms are needed and within what x range these terms may lie.

The analysis techniques mentioned above focus on how well the gap length distribution is being modelled. Another important distribution to be considered is the burst length distribution. The burst length distribution for the model as related to the transition probabilities has been specified in earlier discussion. To verify that the burst characteristics of the channel are being modelled, this distribution can be compared to the observed burst length distribution values.

Often, only gap length statistics are accumulated across a channel. In the absence of burst length data, a rougher comparison can be made between the observed bit error rate across the channel and the theoretical bit error rate calculated from the Markov transition probability matrix. The theoretical bit error rate is calculated as the steady-state probability that the channel will be in the single error state at any given time. This steady-state probability vector for all states ,π^T, can be obtained from the transition probability matrix, P, by making use of the following equations:

$$\pi^T = \pi^T P$$

and

$$\sum_{all j} \pi_j = 1.$$

The theoretical bit error rate would be π^T_n, assuming state n is the e.ror state. This value can be compared to the observed bit error rate of the data communications channel in question to obtain a broad measure of how well the Markov model duplicates the burst characteristics of the original channel. This, along with the comparison measures for gap length characteristics presented above, provide the means for evaluating how well a Markov model simulates the channel it is meant to model.

The method for obtaining Markov models from gap length data and evaluating the resulting models was first used with output from published Markov models as input to the procedure. Markov matrices very similar to those used as input were obtained. Standard deviation values were small, and correlation coefficients were very close to one. When the resulting Markov model had the same number of states as the original input model, the residuals were small. Also, the burst distributions for the original and resulting Markov models were very similar.

The method was also used with data from a "heuristic" model as input. The burst characteristics for the "heuristic" model and the resulting Markov model were very different. More discussion of the problems encountered with the "heuristic" model is included elsewhere in this report.

Some live data collected between Rome Laboratory and Verona, NY, was also used as input to this procedure. For this actual channel data, determining the appropriate number of states necessary in the Markov model was more difficult. As more states were used, the measures related to the gap length characteristics of the channel improved. For example, a three-state model for one data set from this channel produced a standard deviation value of 0.0104 and a correlation coefficient of 0.9905 while a five-state model produced a standard deviation of 0.0029 and a correlation coefficient of 0.9993.

4.0 The Meteor Burst Model

4.1 Review of Meteor Burst Communications

Meteor burst communications is a type of communications based on the reflective properties of the ionized trails emitted by a meteor as it passes through the Earth's atmosphere. Every day nearly 10^{10} meteors pass through the atmospheric layer of the Earth. When a meteor enters the atmosphere, it collides with air molecules producing a trail composed of free electrons and positive ions. When the trail of ionization has an orientation suitable for a given transmitter and receiver, reflection of the radio waves off of the trail can be used for radio communication. The electron density of these particle trails is great enough to reflect frequencies in the VHF range, 30-120MHz.

A meteor trail is categorized according to its electron line density. If the density is low enough that the radio wave penetrates the surface and is reflected from each individual electron within the trail, the trail is defined as underdense. These are trails with an electron density less than 10^{14} electrons/meter. If the electron density of the trail is greater than 10^{14} electrons/meter, the trail is defined as overdense. In this case, the wave does not enter the interior of the trail, but is reflected from its surface. As the trail expands, the resulting multipath causes an exponential decay of the amplitude of the wave.

The type of trail is a factor in the duration of the trail as well as in the arrival rate of meteors. The duration of the underdense trails is much shorter than the overdense trails (an average of 0.5 - 1 second duration for underdense versus 1.5 - 2 seconds duration for overdense trails). However, underdense trails occur much more frequently

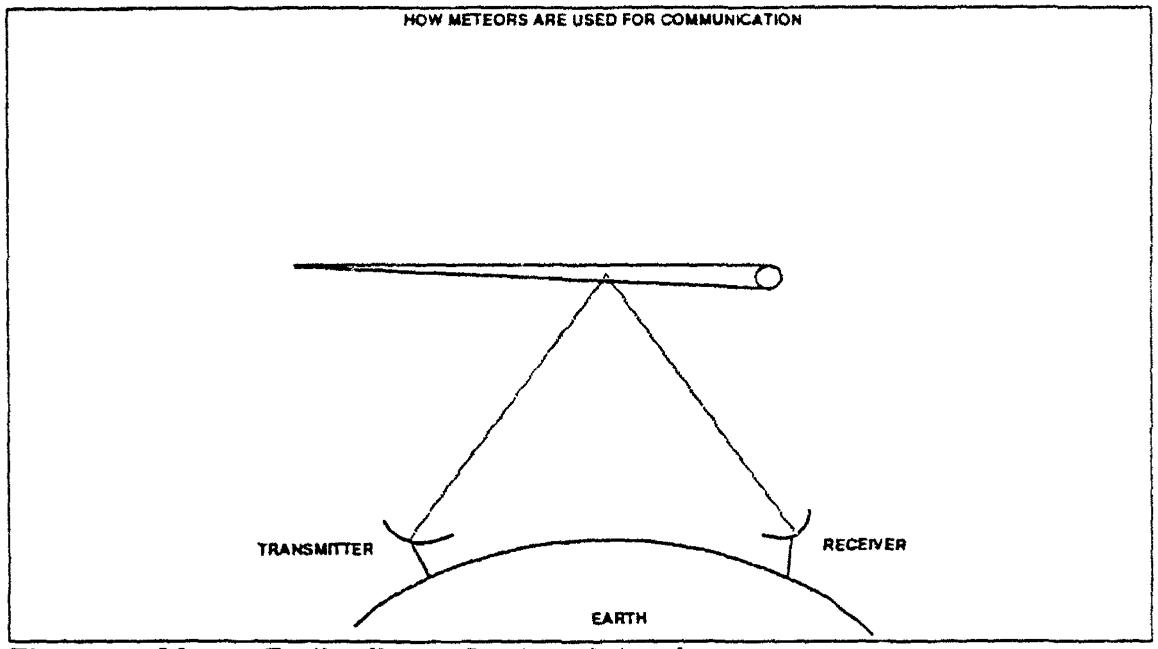

Figure 11. Meteor Trails allow reflection of signals

than overdense trails (less than 2 minute intervals versus 20 minutes or more for overdense) and account for a greater proportion of the throughput on most meteor burst links. For these reasons, many of the meteor burst models use only the underdense trail characteristics when calculating the received power capabilities giving a worst case result. However, at ultra-high frequencies and short path lengths, or for systems that are subject to high levels of man-made noise, overdense bursts can provide the majority of the throughput.

The arrival rate of meteor bursts with suitable orientations for communications is assumed to be a Poisson process. Thus, the interarrival rate has an exponential distribution with a density function of [2]

where a is the mean interarrival time. Solving for the interarrival time:

$$f(t_{ia})=(1/a)*e^{-t/a} \tag{14}$$

$$t_{ia}=-a*\ln(y) \tag{15}$$

where y is a random variable with a uniform distribution.

A default mean interarrival time for the model being constructed is assumed to be 8.5 seconds. This value was obtained by averaging the diurnal and seasonal values for Oetting's work in the area [2]. The diurnal variations cause changes in the mean interarrival time from 4 seconds at 8 a.m. to 20 seconds at 8 p.m. with two hour adjustments in the mean rate. Seasonal variations cause changes from a low of 0.6 times the mean interarrival time in February to a high of 2.25 times the mean interarrival time in June with monthly adjustments.

The duration of the trails has also been shown to be an exponential distribution [3]. As previously mentioned, the electron line density is a large factor in the length of the duration. As the trail expands, an exponential decay of the amplitude of the wave causes an exponential decay in the reflected signal power. The end of a trail is defined as the time when the SNR falls below a given cutoff threshold. Using the Sugar equation [1] for received signal power from an underdense trail, Miller [3] has calculated the duration of the trail as

$$t_d=r*\ln(\frac{Bq^2}{SNR}) \tag{16}$$

Variables r and B are based on system parameters such as transmitter power, orientation of the trail and the height of the trail. For this project, these variables are assumed to be constant. The mean value, r, has been defined to be 1.5 seconds. The default value for the SNR cutoff threshold is 10 dB. The variable q is the electron line density of the meteors and is assumed to be greater than 10^{6} electrons/meter. Due to the falloff rate of overdense trails being different than underdense trails, rather than increasing the complexity of the model, all meteors are assumed to be underdense, which provides the worst case scenario for the model.

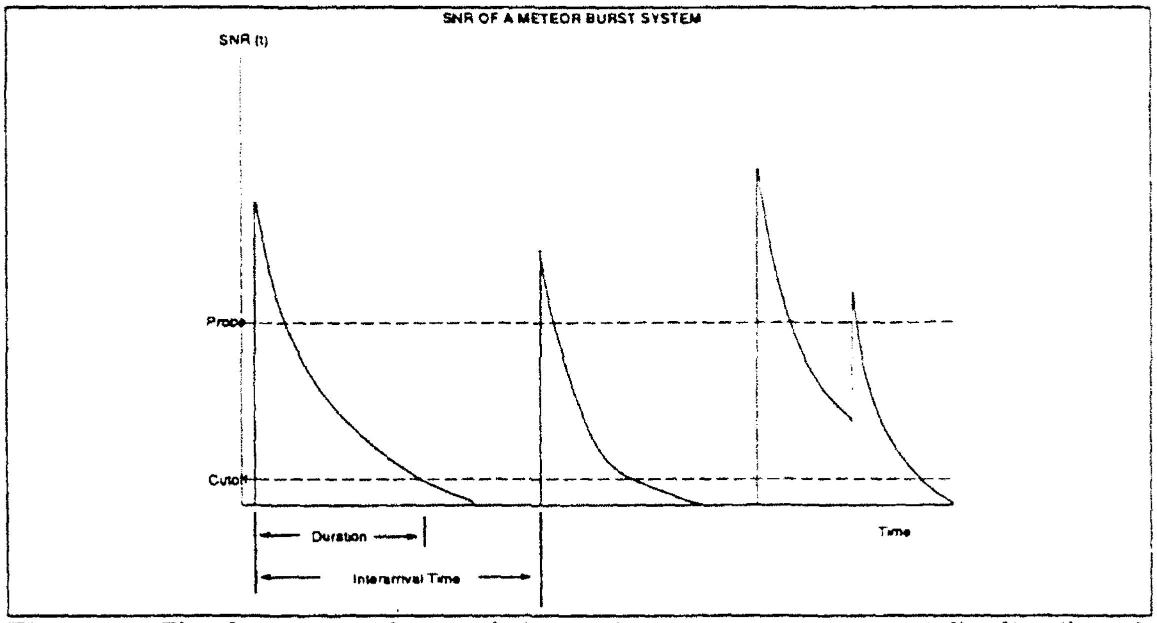

Figure 12. The duration and arrival times of meteors are exponentially distributed. The SNR has an exponential decay.

4.2 Modelling a Meteor Burst System Using Markov Models

Stephen Tsai has shown that an N-state Markov model can be used to describe errors on binary communication channels with memory. By evaluating certain statistics of the channel such as the error free run distribution, or gap length distribution, a

Markov model can be produced. A gap is defined as the number of 0's, or error free bits, bounded on each end by one error bit. The gap length distribution is the probability that gaps of varying lengths will occur. The distribution has been shown to be exponential and can thus be written as a sum of weighted exponential terms [4].

A meteor burst communication channel can be modelled as a discrete time Markov chain. The state space of that Markov chain can be partitioned into two subsets - an operating state and an inoperative state - alternating between the two states, forming an alternating renewal process. The first state is the inoperative or off state where no data is transmitted. This reflects the periods when no meteor burst activity occurs. The second state of the model is the operative or reliable state where data is successfully transmitted with a probability equal to the bit error rate [5].

Note that this model is very different from the theoretical binary communication channel where data is constantly transmitted, the two states represent error-free and error states, and the transition probabilities are the probability of transitioning between the error-free and the error state. The inoperative state in the meteor burst model represents the time periods when no meteor burst is present. For this project, the assumption is made that data is continuously transmitted, so this represents an all error state. Transition probabilities between the inoperative and the operative state are based on probability distributions of a meteor occurring and the probability distribution of the duration of a meteor trail.

In order to model the error transitions in the operative state, the state is further modeled as a sub-stochastic model where the transition probability to an error state is equal to the bit error rate of the system. The number of states in the operative state is dependent on the complexity of the channel, how many factors are allowed to vary in the model, and the assumptions that are made about the given system. For this project, assumptions have been made that the signal-to-noise ratio (SNR) of the system is not constant, but rather has an exponential decay over the lifetime of the trail. It is assumed that the BER of the system changes inversely with the SNR, so decays exponentially from the beginning of the trail.

4.3 The Simulation Model

To accurately determine the error characteristics of a meteor burst channel, real world data from an actual meteor burst channel system would be the ideal approach. Only one system, the Janet System [6], transmitted data without any packetizing of data or using an ARQ form of acknowledgement for correctly received data. Although this data would have been beneficial in the analysis of error characteristics for actual channels, the data was unattainable. Thus, a decision was made to develop a meteor burst channel model using the appropriate equations for factors that affect the BER. The output of the model could produce data with error characteristics that are similar to that of a true meteor burst system that could be used as input to a series of programs developed for producing Markov models from data communications error data. These programs were developed in conjunction with this project.

User inputs into the simulation include the data rate and the amount of time for the program to run. The default data rate is 1200 bits/second. Unit time for the simulation is defined as the amount of time required to transmit one bit or 1/data rate. The program runs while system time is less than the amount of time the program is to run.

When the program is started, system time is reset to zero. A normally distributed random number is generated and then the time of the first meteor arrival is determined by the equation

$$a\_time = -a\_mean * \log(rn) \tag{17}$$

where *a_mean* is the mean interarrival time (8.5 seconds) and *rn* is the random number generated. The default mean interarrival time for the model, 8.5 seconds, is based on Oetting's work in the area and assumes a cutoff threshold of 10 dB.

The duration of the trail is determined in much the same way. A random number is generated and the duration time is calculated as

$$d\_time = -d\_mean * \log(rn) \tag{18}$$

where *d_mean* is the mean duration time (1.5 seconds) and *rn* is the random number generated. The default mean value of 1.5 seconds is based on data obtained from the Comet system.

The system clock monitors the time of the simulation with new meteors being generated every time a meteor is activated. This allows for multiple meteors to be active concurrently.

For the simulation, we assume a strict channel model. Each bit is only transmitted once and no error correction is done on any data. Transmission is continuous with no probing information or packetizing of data. Because no form of error correction coding or multiphase shift keying is being utilized, the assumption was made that binary decisions were only based on optimum detection of equal-energy orthogonal signals in white Gaussian noise. Because of this assumption, we can determine the error probability strictly from the total received signal energy and the noise strength [7]. The SNR for any part of the channel is thus a lower bound estimate of success probability.

The BER of the received signal is defined as inversely proportional to the time varying nature of the SNR over the lifetime of the trail. The probability of error for a bit when the meteor trail is active can be written as:

$$p_e = 1 - e^{\left[\frac{(sys_{time} - on_{time})}{duration_{time}}\right]} \tag{19}$$

where sys_{time} is the present simulation run time, on_{time} is the start time of the burst, and $duration_{time}$ is the amount of time the trail will be operational. A normally distributed random variable is generated and compared to the probability p_e. If the value of the random variable is greater than p_e, an error is said to occur.

One hundred percent errors are assumed to occur when no meteor activity is present. The probability of error during the lifetime of a trail is inversely proportional to the SNR of the signal. An assumption is made that all trails start off with a probability of error equal to zero and exponentially increase until the cutoff threshold is reached.

The output consists of 0 for non-error bits and 1 for error bits. A run length encoding scheme is utilized for compression as the data is saved. When defining the compression scheme, several assumptions were made. Data was not to be collected in the inoperative state as this data contains all errors, except to track the number of time units. During the operative state, the BER is assumed to be low, so the data is assumed to be primarily non-error bits, or 0's. A count of the number of non-error bits is kept until an error, or 1, occurs. At this point, the number of non-error bits is recorded followed by ":", followed by the number of errors (usually 1) in an external file. The recording of the error bit was necessary to track the bits at the end of trail.

The model was written in "c" on a Sun workstation using Drand48 as the random number generator. This generator was chosen for the calculations as it seemed to provide a fairly normal distribution, passing a Chi-square test with 10 degrees of freedom.

4.4 Analysis of Data from Simulation

A 200 second run was used to analyze the data output from the simulation. This run length provided over 10^6 data bits which were collected in compressed form. This gap data was then used as input to the Markov model production program. The process produced a list of gap lengths and a best fit to the exponential equation. The coefficients and exponents from this equation were used to produce a Markov model. The Markov model mapped to the gap length distribution very well, but, the resulting Markov model did not fully represent the nature of the errors. In the original data, the distribution of gap lengths was exponential during the life of a trail. Thus, the probability of longer gaps was much higher at the beginning of the trail and decreased as the trail decayed. Having a gap length distribution only determines the number of gaps of each length and does not take into consideration time dependencies of the data. In Tsai's paper, the distribution of gaps over time had a uniform distribution, whereas with meteor burst systems, the distribution of gaps over time is exponential.

It was shown that in order to produce this exponential change in gap distribution, a series of Markov models must be used with each matrix having individually corresponding gap error characteristics. Since the distribution is constantly changing, ideally an infinite number of Markov models would be needed to represent the change.

A series of Markov models was chosen as a possible way to model the exponential change in gap distribution. The data generated for each trail by the simulation was segmented into seven parts. Although this is not a true representation of the exact time varying nature of the BER and graininess occurs due to the limited number of segments, this is a way to try to interpret the time varying nature of the BER. Each segment has uniform length along the life of a trail. Analysis of each Markov model derived from the segmented data seems to indicate that a different segmentation scheme might be more meaningful. Due to the exponential nature of the variation in BER, most of the change occurs during the first three segments of the trail. To more closely approximate this, shorter segment lengths at the beginning of trail would allow a closer evaluation of each segment.

4.5 Conclusions

Modelling a meteor burst communication system requires intricate knowledge of the details of the system, which often are not available. Many of the factors involved must be assumed to not affect the distributions used in modelling. Without real world data, true representations cannot be verified. The model represented in this paper has shown that a single Markov model cannot be used to represent the burst error characteristics of a meteor burst system without sub-stochastic models being imbedded within the model.

5.0 N-Bit Bursts Over M-Percent Distribution

5.1 General Introduction

The idea of establishing some heuristic assumptions about the distribution of errors within a given bit error rate has some appeal for use with the Error Injector Units (EIU). That is, if the overall bit error rate (BER) is known, one might be able to hypothesize the statistical distribution of errors that might occur within the burst, once the burst has begun. While it is unlikely that such distributions would model any real world data channel exactly, it somehow seems intuitive that such patterns are more realistic than simple Gaussian models.

As an example, assume that a standard RS-232 data channel has an known overall error rate of 10^{-3}. Further assume that we have reason to expect that the errors tend to occur mostly in groups of two, with some single errors, and some three bit errors. The problem is then to design a Markov model that will simulate an error distribution of this sort.

5.2 Building the First Model

In discussing the burst error distribution, we will assume the standard error model of:

$$y_i = x_i + e_i$$

where {x} represents the bit string of the original transmitted data. The data channel corrupts the incoming data with interference and produces the output data string {y}. The string {e} is the error string. A one in the string {e} represents an error, while a zero represents the absence of an error. The "+" operator represents modulo 2 addition or an exclusive or operation.

Continuing the earlier example, with a BER of 10^{-3}, let us assume an error distribution that favors two bit errors (50%) versus single bit errors (25%) or three bit errors (25%). The beginnings of this model are shown in Figure 13. There is a single non-error state. The model would remain in this state with a probability of 1-BER. There would be a probability of BER that the model would transition to <u>some</u> error state. The transition to each error state is governed by the relative probability of an error of each type and the BER. The probability of a transition to the error state where i error bits are produced is labeled p_i in Figure 13.

In computing the probability of transitioning to each error state, the number of error bits generated in this state must be considered in order that the overall BER not be exceeded. To make this point clear, consider a simple model that produces either one or two bit error bursts. If the two error bursts are equally likely, one would be tempted to set the transition probability to BER/2 for both single and double error bits.

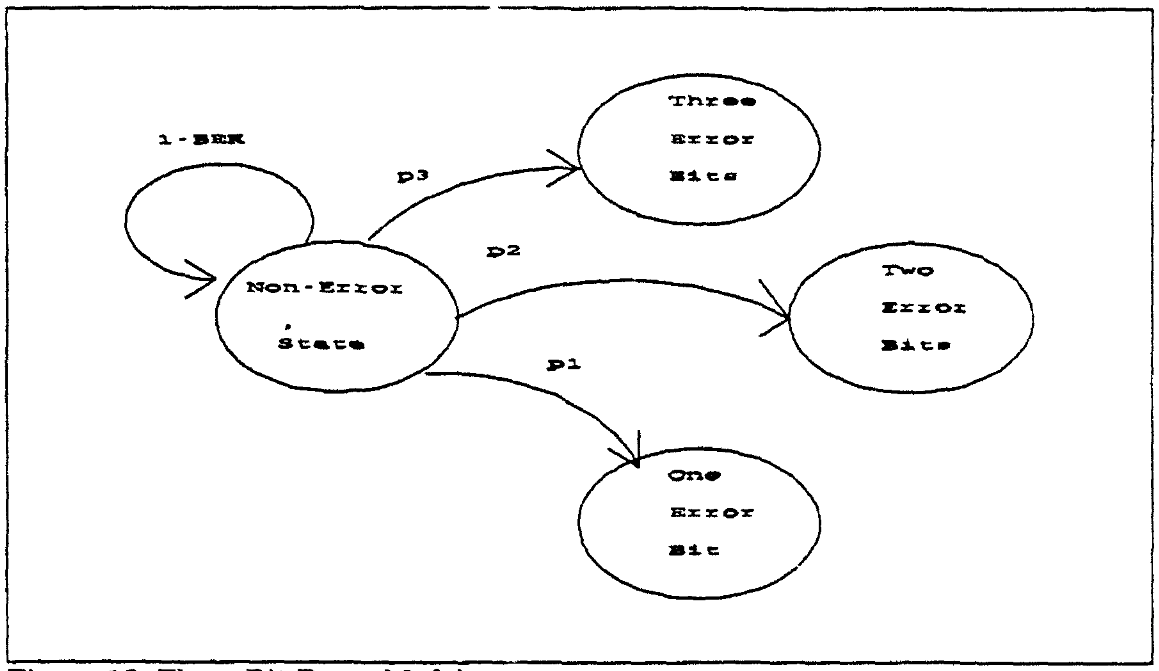

Figure 13. Three Bit Burst Model

Unfortunately, this does not work. Such an assignment would result in transitions to an error state at a rate equal to BER, as would be appropriate, but half the time, the error state would produce *two* error bits. This means that for each two error transitions (i.e. one BER probability) we would get an average of three error bits. This would result in an actual error rate of 1.5 BER. This is clearly unacceptable.

The probability of transitioning to each error state must be adjusted so that while 50% of the errors are indeed two bit errors, the overall BER must remain constant. These goals can be achieved by computing the probabilities of transitioning to each error state so as to take all these factors into consideration. The process proceeds as follows (assuming that BER is known).

First, we define the relative probabilities for a 1, 2, or 3 bit error. In the example in question, this might be:

$rp_1 = .25$
$rp_2 = .5$
$rp_3 = .25$

Now we determine the relative ratios of errors of each type. To do this, we divide the relative probabilities of each error type by the smallest error probability. If two error probabilities are equal, as in our case, we use either smaller value as the divisor. From this computation, we obtain R_i, which is the ratio of occurrences of 1, 2 and 3 bit errors:

$R_1 = rp_1/rp_1 = 0.25/0.25 = 1.0$

$R_2 = rp_2/rp_1 = 0.50/0.25 = 2.0$

$R_3 = rp_3/rp_1 = 0.25/0.25 = 1.0$

Now we must assume a BER distribution of errors that also meets the requirements of the p_i statistics above.

If we had a perfectly distributed group of errors, we would expect one three-bit error burst, two two-bit error bursts, and one single-bit error burst in a complete set of errors. To simplify the discussion, it is convenient to define the total number of error bits that occur in one perfectly distributed set (EBIOS) as:

$$\text{EBIOS} = \sum^{n} i(R_i)$$

$$i=1$$

where n is the number of bits assumed in each error burst.

Now, the total number of data bits (TNDB) over which this perfect distribution will take place must be such that the overall BER remains constant. This leads to the formula:

$$\text{BER} = \text{EBIOS/TNDB}$$

or $$\text{TNDB} = \text{EBIOS/BER}$$

Continuing with the example from above, the EBIOS would be computed as:

$$\text{EBIOS} = \sum_{i=1}^{3} i(R_i) = 1\text{x}1 + 2\text{x}2 + 3\text{x}1 = 8$$

This value is easily explained by recalling that there will be one error burst that produces a single error bit, two error bursts that produce two error bits, and one burst that produces three error bits. Hence, there will be eight errors in one perfectly distributed set.

This ratio of errors will retain the desired relative probabilities, but in order to maintain the proper BER rate, these errors must be distributed over the appropriate number of data bits. In our example, number of bits over which this is distributed is computed as:

$$\text{TNDB} = \text{EBIOS/BER} = 8/10^{-3} = 8 \times 10^{3} = 8{,}000 \text{ bits}$$

With these values in hand, it becomes possible to compute the probability of transitioning to each of the three different error states. These values are computed from the formula:

$$p_i = R_i / \text{TNDB}.$$

For the example, these values compute to:

$$p_1 = R_1 / \text{TNDB} = 1/8{,}000 = 1.25 \times 10^{-4}$$
$$p_2 = R_2 / \text{TNDB} = 2/8{,}000 = 2.50 \times 10^{-4}$$
$$p_3 = R_3 / \text{TNDB} = 1/8{,}000 = 1.25 \times 10^{-4}$$

These values will be used later to construct a Markov model that will produce the desired error distribution.

It would be instructive to utilize these computed probabilities to recompute the overall BER for the entire model just to verify that the BER is as intended. With a model such as the one under discussion, the overall BER is given by the formula:

$$\begin{aligned} \text{BER} &= \sum_{i=1}^{n} i(p_i) = \\ &= 1(1.25\text{x}10^{-4}) + 2(2.5\text{x}10^{-4}) + 3(1.25\text{x}10^{-4}) \\ &= (1.25 + 5 + 3.75) \text{ x } 10^{-4} \\ &= 10^{-3} \end{aligned}$$

Since this value is equal to the original BER, we may assume that the mathematics are correct.

The model shown in Figure 13 is somewhat oversimplified. The nature of Markov models is such that each state can produce only one output. In our case, that is either an error (1) or a non-error (0). In Figure 13, two of the states indicate the production of two or three bits of output. Since this is not consistent with the Markov model, these states must be replaced with a sequence of states, each of which produces a single bit of output.

Because the errors may come in bursts of up to three bits, the transition to an error state may consist of a *sequence* of states that will produce the error string desired. That is, in Figure 13, the state labeled three error bits must actually consist of three states in sequence, each of which produces a single error bit. In addition, the state labeled two error bits would similarly be two states each of which produces a single error bit.

The transition from the non-error state is controlled by p_i as computed above. Once a particular error sequence has started, however, the transition to the next state of the sequence is pre-determined. That is, once a two or three error bit sequence begins, the cycle continues through all the states required to produce the correct number of error bits, before returning to the non-error state.

Fortunately, it is possible to combine the error states described above. If we think of the three error bit sequence as being the basic string, it is possible to utilize part of this string for the one and two bit error sequences. That is, the one bit error state is simply the last state of the three bit error sequence. The two bit error state is the last two states

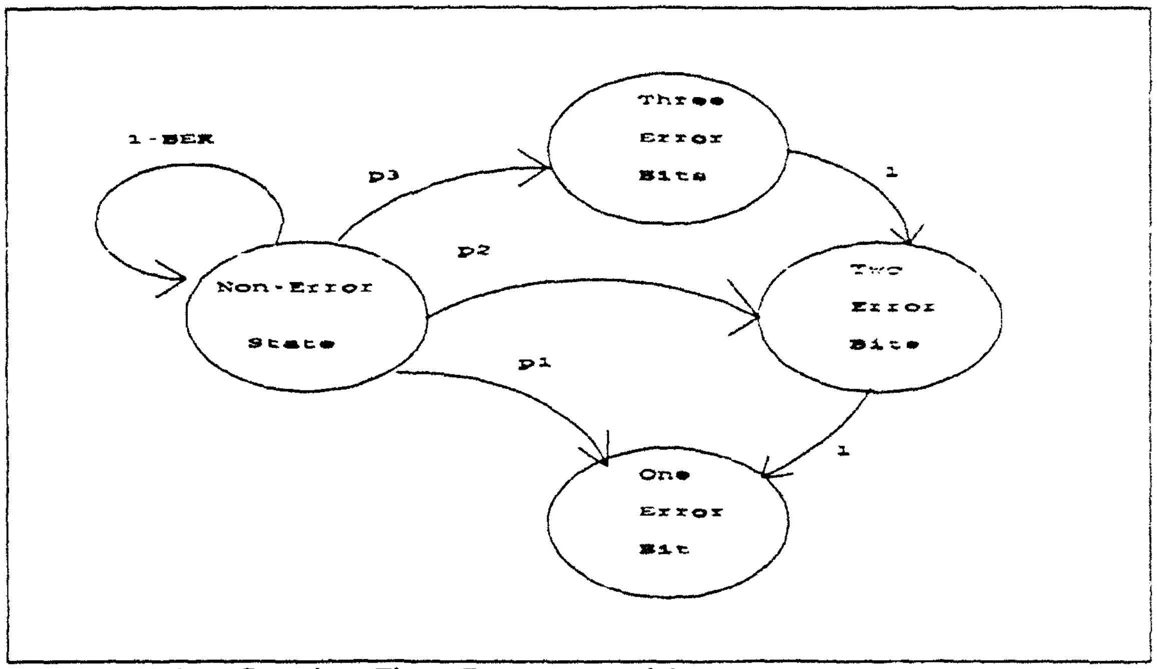

Figure 14. More Complete Three Bit Error Model

of the three bit error sequence. In fact, it is relatively easy to modify the model from Figure 13 so as to provide the needed error bit generations. This revised model is shown in Figure 14. Note that the transition from the three-bit state to the two-bit state, and from the two-bit state to the one-bit state both carry a probability of 1. This insures the correct sequence of states required to produce the required number of error bits in the burst.

The final parameter missing from the model in Figure 14 is a transition to the appropriate state at the end of the error burst. One possibility would be to transition to the appropriate error or non-error states based on the same probability as in the non-error state. This at first seems reasonable, since the next state would normally be determined stochastically. However, a little reflection reveals that this would be an inappropriate transition. The problem created by such a transition is that it would permit a second burst to immediately follow the first. If this is allowed to happen, then two three bit bursts could be interpreted as a single six-bit burst. This is clearly not desirable in a model that is intended to produce bursts with a maximum length of three bits.

To prevent the generation of bursts that are longer than desired, it is necessary to insure that there is always at least one zero bit generated after a burst has been generated. As we shall see later, the actual number of zero bits required is a function of several variables, but there will always be at least one zero bit after each burst.

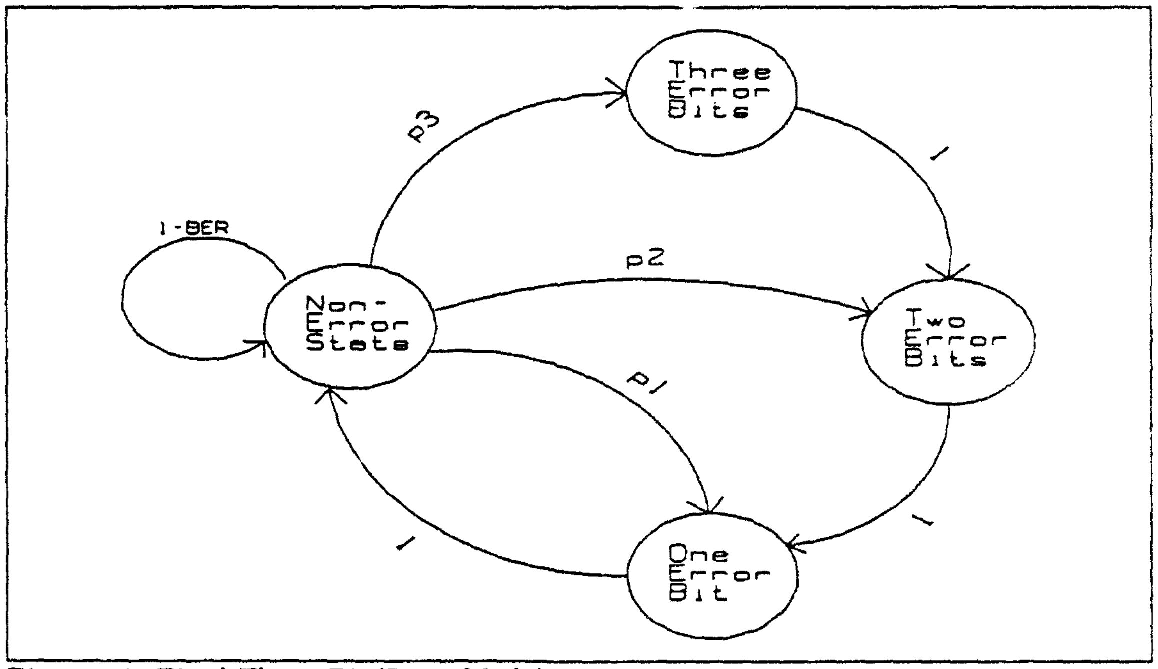

Figure 15: Final Three-Bit Error Model.

Therefore, in the case in question, the appropriate transition from state 2 will always be back to state zero. This will insure the generation of at least one zero bit after the generation of any burst. This final state transition is indicated in Figure 15.

Given the procedure above, the parameters for the heuristic Markov transition table are easily computed. The table for the sample problem from this discussion can be constructed using the probabilities computed in the preceding section. The resultant values are given in Table 1. State 1 is the non-error state, and states 2 through 4 produce 1 through 3 errors respectively.

Probability of transitioning to state:

Current State	1 non-error	2 one error	3 two errors	4 three errors
1	0.9995	0.000125	0.000250	0.000125
2	1	0	0	0
3	0	1	0	0
4	0	0	1	0

Table 1: Markov Table for Example

This specific table can be generalized to include any specific number of error bits in a burst, any BER, and any desired relative probability between bits. The p_i values computed from the formulas above will make up the first row of the table as the probabilities of transitioning from the non-error state to any one of the error states. The value 1-BER will be inserted into the first table position. In this table, n would represent the maximum error burst size and can be extended to any desired size.

The ease of constructing this type of Markov table lends itself readily to computer implementation. For purposes of this project, a program was written in the "c" language to perform the computations discussed above. This program accepts input values for the BER, the maximum number of error bits (n), and the relative error probabilities for 1, 2, ..., n error bits. The output from the program is a Markov table in a format similar to that given in Table 2 that satisfies the given parameters.

Probability of transitioning to state:

Current State	1 non-error	2 one error		n n-1 errors	n+1 n errors
1	1 - BER	p_1	o	p_{n-1}	p_n
2	1 - BER	p_1		p_{n-1}	p_n
3	0	1		0	0
	o		o		o
n	0	0		0	0
n+1	0	0	o	1	0

Table 2: Generalized Markov Table

In order to test the output produced by the Markov table generation program, a second program was written that utilizes the Markov table from the first program to run a computer simulation of the errors that would be produced by this model. The results from the second program indicate that the Markov tables produced in the first program do, in fact, realistically model the hypothesized error performance.

5.3. Shortcomings of this Model

There are two major shortcomings with the model just described. In order to understand these shortcomings, a more precise definition of an error burst is required. Generally speaking, an error burst is considered to be a set of error and non-error bits that occur together in groups. By definition, the first and last bits within the burst are always a 1, representing an error. Between the beginning and ending error bits of the burst, there will be n-2 "fill" bits that may be either error bits or non-error bits.

Within the burst, the error bit density, or Δ_0, is defined as:

$$\Delta_0 = \text{number of error bits/burst length}$$

The Δ_0 value is used to define the limits of a burst. Without a definition of Δ_0, it is impossible to determine whether the error string "...10001..." represents a single burst with five bits (two error and three non-error) or two bursts, each with a single error bit separated by three non-error bits. In this particular example, if $\Delta_0 <= 0.4$, then the string in question represents a single burst; otherwise it represents two separate errors.

Considering this definition of a burst, it becomes obvious that the proposed model suffers from a lack of flexibility pertaining to Δ_0. This model always produced a burst in which all the bits were error bits. In other words, Δ_0 was always assumed to be one. Hence, no non-error bits ever occurred in the burst. This prevents generation of four-bit error bursts such as 1101, 1011, 1001, etc. On real world data channels, it appears that most error bursts do contain some correct data bits. Frequently up to 50% of the bits within the burst may be non-error bits.

Markov models for producing bursts containing both errors and non-error bits are reasonably easy to design, and the earlier model could be extended to cover this situation. However, as the number of combinations of n-bit errors increases, the number of states required in the model increases proportionally. For example, to generate four-bit bursts in the earlier model requires only five states: the non-error state and the four different error states.

If we wish to include the generation of non-error bits in the burst, we must increase the number of states in the model. In the case of the four-bit model, two additional states must be added to account for the possibility that the center two-bits may be either zero or one.

In a general n-bit burst model there must be three fixed states; one for the start state and one each for the first and last error bits. Between the first and last bits, there will be n-2 bits in the burst. The model must provide for the possibility that either of these can produce zero or one. This requires an additional n-2 states. Hence, the total number of states required is:

$$\text{States} = 2(n-2) + 3 = 2n-1$$

For n=5, the number of states required is 9.

Even this number of states exceeds the number available in the EIU, but the situation gets even worse. Following any burst of length n, there must be a minimum number of non-error (MNZ) bits generated to insure that the second burst is not interpreted as continuation of the first burst. The number of zeroes required is dependent upon the number of error bits in the burst and the value of Δ_0 chosen in the definition of the burst. The mathematical formula used to determine the minimum length of the "gap" between any two bursts can be computed from:

$$MNZ > (int)\ (error\_bits+1)/\Delta_0 - (bits\_in\_burst + 1)$$

Notice that the number of non-error bits that must follow the burst will vary depending on the number of error bits within the burst. If, for example, we use a $\Delta_0 = 0.5$ with a four-bit burst, there will have to be six non-error bits following a burst of "1111", but only two non-error bits must follow the burst "1001".

Using the formula for MNZ, the early model could be modified to produce the appropriate number of zeroes following each burst. Since each zero generated requires another state, the appropriate number of zero-generating states is affixed to the last state of the model, and the problem is largely solved. Determining the state transitions for these new states is fairly complex, since it depends on both Δ_0 and the number of error bits generated. There is, however, a practical consideration that is far more significant than the complexity of computing the state transition probabilities.

The more significant problem in expanding the earlier heuristic model is that either of the modifications discussed above will add a significant number of states to the finite state model represented by the Markov table. Since the Error Injector Units on which this model is intended to run are restricted to eight states, it is obvious that the heuristic model described earlier is severely limited. The maximum number of bits that can be produced with $\Delta_0 = 1$ is seven. If Δ_0 is less than one, then the number of bits permitted in the burst must be further restricted, and the state transitions to accomplish the exac⁺ required number of fill bits becomes more complex.

5.4 Efforts toward an Improved Heuristic Model

Based on the shortcomings discussed above, it was decided to try another approach to building a heuristic model. It was felt that this approach could build on the other efforts in the project by using the data reduction procedures for real data to reduce data from a non-Markov model that would produce error burst of the appropriate length and characteristics. This approach involved the following four steps:

1. Develop a deterministic non-Markov model that will generate error data with the desired characteristics.
2. Use this model to produce error strings that conform to the desired patterns.
3. Use the techniques currently under development for raw error data reduction to reduce the error strings produced from the non-Markov model to the format required for the production of an equivalent Markov model.
4. Verify that the Markov model so produced generates the error strings of the correct type.

Using the approach outlined above, the development of a non-Markov model for generating heuristic error data was undertaken. Some of the procedures for computing state transitions used by the earlier Markov model were incorporated into this model. Other techniques, particularly with reference to the generation on bursts with $\Delta_0<1$ had to be developed specifically to support this approach. In addition, some support programs for testing and analyzing output from the model had to be developed.

One step in this process involves computing the minimum number of data bits (MNDB) that would include the minimum burst set (MBS), and still retain the correct overall BER. This process becomes somewhat more complex when longer burst lengths are included. Consider the case for burst lengths of up to three bits, with one, two, and three-bit bursts being equally probable. The complete error burst set now includes four elements, 1, 11, 101, 111, because there are two different three-bit errors that must be included in the set. But, since one and two-bit errors must be equally probable with the three-bit errors, and since there are two three-bit errors, each of the one and two-bit errors must also occur twice. Hence the MBS becomes {1, 1, 11, 11, 101, 111}. This set contains eleven error bits. Therefore, the minimum number of bits required in one set is 11/.001 = 11000. From this value, the absolute probability of the occurrence of an error burst of each length can be computed.

All these relatively intuitive ideas can be incorporated into a computer program that can perform these computations relatively easily. It is not really necessary to compute the MBS, only the relative number of occurrences of the bursts of different lengths. To compute this relative frequency of bursts of different lengths (r[i]), it is only necessary to determine the burst length with the smallest relative probability (minimum(rp)). Since this burst length will occur only once in the MBS, it should have a r[i] value of 1. By definition, the other burst lengths will occur at least as many times at the burst length representing minimum(rp). The exact ratio of the other burst lengths can be computed from the formula:

r[i] = rp[i]/minimum(r)

Using these values will insure that the appropriate ratio of occurrences of the various length of data bursts will occur in the MBS.

Another problem that had to be addressed in this model was computing the number of fill bits that had to be generated after each burst. After the requisite number of burst bits had been generated, the model also had to generate the minimum number of zero bits immediately following the burst to insure that the generated burst would not be combined with some subsequent burst to form a burst longer that the *specified length*. The minimum number of zeroes was computed based on the number of error bits within the burst and the desired value of Δ_0 as discussed earlier.

The mathematics for the heuristic model were determined, and a computer program (hcomp.c) was written in the "c" language that embodies these mathematical parameters. This program was then used to generate the table of values needed to produce the relative probabilities needed for the burst generation model. This program was tested and performed as expected.

An initial "test of concept" program (heurwork.c) was written and used to accept the output from hcomp.c and to produce error strings based on the input parameters used in hcomp.c. The output from heurwork.c was piped to two other programs written to validate the output from this model. A program called burstlist.c was used to list the bursts as they occured within the output stream from hcomp.c. This program was somewhat limited, since the BER must be relatively high in order to produce enough data for visual validation. The program produced the expected results.

Another program called burstcount.c was used to count the number of bursts of different lengths and types. This program was appropriate for examining the model output for lower error rates and longer strings. Preliminary results were also positive when burstlist was used to analyze the output of the model.

Finally, the output from the second heuristic model was fed to the gapcount program, and the gap length data collected in a manner similar to that used when testing the validity of a Markov model. The gap lengths from this model were then fed into the data reduction procedure in an attempt to build a Markov model that would match the performance of the second heuristic model. The output from this Markov model was collected and examined, and at this point, a significant problem was discovered. The output from the Markov model did not resemble that of the original heuristic model.

When the data produced by the heuristic model was analyzed in gap data format, it became obvious that the gap data produced by this model was not at all like the data that had been collected from the real-world data communications channels. The real world gap data fits an exponential distribution. Because of the finite limitation on the number of error bits in a burst, the gap data from the heuristic model is very different.

When the data from the heuristic program was used in the curve fitting program, the procedure did produce the best exponential fit to the data possible. Due to the fact that the heuristic gap data was not truly exponential in form, the resulting fit was quite poor, and the resulting Markov model did not accurately model the performance of the heuristic model.

In retrospect, it could have been anticipated that this approach would not work. An analysis of the graphs of the gap data from the heuristic model clearly indicates that the data is not in exponential format. Unfortunately, it was necessary to invest considerable time and effort into developing the heuristic model before a comparison of the two outputs was possible.

It now appears that an important point was overlooked in the initial development of the non-Markov heuristic model. This point is that while all Markov models are stochastic processes, not all stochastic processes can be modeled with a Markov model. The Markov property for a stochastic process requires that the next state determination in the Markov model be uniquely determined based only upon the current state of the model, and not on a history of any other states that may have been passed through in order to get to the current state. Unfortunately, this property is violated in the n-bit burst over m-percent distribution model.

To understand why this is so, consider a model that is generating bursts with lengths of either one, two, or three bits. If the model is in a state that represents one of the states within the generation of a burst (i.e., an error state), the next state transition cannot be determined without some knowledge of the preceding states. If the state that the model is in represents the first bit of a burst, then there is a finite, non-zero probability that the next state will also be an error state. On the other hand, if the state that the model is in represents the third bit of an error burst, the probability of a transition to a subsequent error state must be zero, or the model may produce a burst of length longer than three bits.

Hence, a model that produces a burst with a fixed limit on the maximum length of the burst will have the next state determined by a combination of the current state and a history of the states that the model has passed through in order to arrive at this state. This violates the Markov property and would seem to preclude a general Markov model that will produce n-bit bursts over m-percent distributions.

This concept was reenforced by some results obtained from analyzing the output of the Tsai HF Markov model which was run on the Markov simulation program, rawmarkov.c. The output from this model was piped to the burstcount program and produced some very interesting results. While most of the error bursts tended to be in the range of just a few bits, usually from five to ten bits, this model frequently produced error bursts with lengths of greater than 20 bits.

What this means is that in the Markov models that we have been using to represent real world data communications channels, there is no upper limit on the length of the burst that may be produced. Regardless of the number of error bits that have already been produced, the probability that the next bit will be another error bit will always be greater than zero. These results would seem to indicate that the concept of a n-bit burst over m-percent distributions may not be representative of the characteristics of a real data communications channel.

5.5 Conclusions

The major results of these limitations are that the original model is restricted to $n <= 7$ and $\Delta_0=1$. While not as general as we had hoped, this model may be of some value in testing data communications protocols where errors may be assumed to be of this form. It is, unfortunately, not clear exactly where such error patterns might occur.

The original model can also be modified slightly to produce some error bursts with non-error bits in them. For example, the model from Table 1 could easily be modified to produce error strings of the form "101" by the addition of a new state 3a. State 3a would be a non-error state, and the transitions from state 4 would be modified to 0.5 probability of a transition to 3 and a 0.5 probability of transitioning to state 3a. Hence, half the time, the three bit burst generated by this model would be of the form "111" and half the time would be of the form "101". A minor adjustment would have to be made in row one of the table to take into account that the that two transitions to state 4 would produce an average of five error bits, but that is relatively easy to fix. Unfortunately, if we attempt to extend this technique for four or more error bits, the number of states needed then exceeds the number available in the EIU, so the technique hardly seems worth the effort.

All things considered, the original model probably pushes the n-bit error out of m-percent distributions concept about as far as it will go with the eight state limitation of the EIU. Further, any attempt to pursue the n-bits errors over m-percent distribution model now appears far less attractive than it did at the beginning of this project. An examination of the output from the non-Markov heuristic model which performed precisely as it was programmed to do, reveals that it has little in common with the error characteristics that we have been able to examine from the real world data communications channels. While there is always the possibility that further research in this area may bear fruit, it now seems unlikely that such a model would be easy to develop or very useful when developed.

6.0 Summary

In summary, the project was a limited success. The technique and procedures for building a Markov model from the error gap data were quite successful. A computer program to perform the required curve fitting was identified, and was found to be quite satisfactory. Good progress was made in determining a method for making the initial guesses for the non-linear regression parameters required for the curve-fitting program. In addition, the technique for evaluating the accuracy of the Markov model in representing the original error gap data proved to be satisfactory. Further work in the accuracy measurements will continue with Ms. Burns' Masters Thesis, and the results are expected to confirm the conclusions from this report.

Significant work in the area of the meteor burst Markov model was also accomplished. Unfortunately, the time dependent nature of the error rate characteristics make it very difficult to develop a single Markov model that will effectively mimic the channel error characteristics of a meteor burst communications channel. As an alternative, the use of multiple Markov models that are stored in the EIU and swapped on a time dependent basic will give a workable approximation of the characteristics of such a channel. Using eight Markov models that represent the varying phases of a meteor burst channel communication capabilities will give a reasonable approximation of the meteor burst channel.

The N-bit burst over M-percent distribution Markov model suffers from the limited number of states available in the EIU. With only eight states, models with error bursts of up to seven bits are possible, with differing probabilities for error bursts of one, two, ... seven bits are possible. This model is limited to a Δ_0 value of 1.0. Models with shorter bursts and $\Delta_0 < 1$ are possible but are quite limited as to burst length and/or Δ_0.

Overall, the best technique for modeling data communications errors remains one of collecting error gap data from live transmissions, and then using the techniques proposed in this report to reduce that data to an appropriate Markov model. With an adequate quantity of data collected, it should be possible to model enough different Markov models to make the Error Injector Units a viable vehicle for developing robust protocols over a wide variety of data communications channels.

7.0 Bibliography

[1] G. R. Sugar, "Radio propagation by reflection from meteor trails," *Proc. IRE*, pp. 116-136, Feb. 1964.

[2] J. D. Oetting, "An analysis of meteor burst communications for military applications", *IEEE Trans. Commun.*, vol. COM-28, no. 9, pp. 1591-1601, Sept. 1980.

[3] S. L. Miller and L. B. Milstein, "A comparison of protocols for a meteor-burst channel based on a time-varying channel model", *IEEE Trans. Commun.*, vol. 37, no. 1, pp. 18-30, Jan. 1989.

[4] S. Tsai, "Markov characterization of the HF channel", *IEEE Trans. Commun.*, vol. COM-17, no. 1, pp. 24-32, Feb. 1969.

[5] Y. Chandramouli, M. F. Neuts, and V. Ramaswami, "A queueing model for meteor burst packet communication systems", *IEEE Trans. Commun.*, vol. 37, no. 10, pp. 1024-1031, Oct. 1989.

[6] P. A. Forsyth, E. L. Vogan, D. R. Hansen, and C. O. Hines, "The principles of JANET-A meteor-burst communication system", *Proc. IEEE,* Dec. 1957.

[7] J. J. Metzner, "Improved coding strategies for meteor burst communication", *IEEE Trans. Commun.*, vol. 38, no. 2, pp. 133-136, Feb. 1990.

[8] B. D. Fritchman, "A binary channel characterization using partitioned Markov chains", *IEEE Trans. Inform. Theory,* vol. IT-13, pp. 221-227, Apr. 1967.

[9] Daniel Cuthbert and Fred S. Wood. *Fitting Equations to Data*. New York: John Wiley and Sons, Incorporated, 1980.

[10] E. O. Elliott, "A Model of the Switched Telephone Network for Data Communications." *The Bell System Technical Journal 46*, no. 1 (1965): 89-109.

[11] Gilbert, E. N. "Capacity of a Burst-Noise Channel." *The Bell System Technical Journal 39*, no. 9 (1960): 1253-1265.

[12] Isaacson, Dean L., and Richard W. Madsen. *Markov Chains, Theory and Applications*. New York: John Wiley & Sons, Incorporated, 1976.

[13] Larson, Roland E. *Elementary Linear Algebra.* Lexington, MA: D. C. Heath and Company, 1988.

[14] McCullough, Richard H. "The Binary Regenerative Channel." *The Bell System Technical Journal 47*, no. 10 (1968): 1713-1735.

[15] Miller, Irwin, and John E. Freund. *Probability and Statistics for Engineers.* London: Prentice-Hall, Incorporated, 1990.

[16] Shanmugan, K. Sam and A. M. Breipohl. *Random Signals: Detection, Estimation and Data Analysis.* New York: John Wiley and Sons, Incorporated, 1988.

[17] Stallings, William. *Data and Computer Communications.* New York: Macmillan Publishing Company, 1991.

[18] Varshney, Pramod K. "Channel Models for the Error Injector Unit." Rome Air Development Center Final Technical Report. RADC-TR-90-89. May 1990.