

# **A Four-State Trellis-Coded 8-PSK Modulation Computer Simulation**

**BY**

**BRIAN KOPP  
ELECTRICAL ENGINEERING  
NEW MEXICO STATE UNIVERSITY**

**SPRING 1988**

## **ABSTRACT**

The continuing growth of the telecommunications industry has created a steadily increasing need for higher performance communications systems - systems that can transfer data at faster rates while meeting stringent bit error rate requirements. In the case of satellite and mobile communications these same systems must also maintain minimum size and power consumption requirements. To help implement this industry demand computer simulations of communications systems can be a viable tool. Simulators can be used to demonstrate feasibility while maintaining minimum research and development costs during the design phase of these new and more complex communications systems.

One type of system where simulation has proved helpful has been trellis-coded-modulation (TCM). This paper documents a simulation of a four-state trellis-coded eight-PSK modulation scheme currently being researched at New Mexico State University (NMSU). In the past simulations of convolutionally coded schemes have used binary symbols in the decoding process. In TCM the Euclidean space components of the modulation scheme are used in place of the binary symbols. The simulator under development incorporates these Euclidean signal components which are taken from an eight-PSK signal constellation. Soft-decision maximum likelihood decoding using trellis trace-back techniques are then applied to the Euclidean signal components to recover the simulated transmitted data. The simulator supplies the user with the number of undetected errors generated during a simulation as well as the bit error rate for a given signal to noise ratio.

This simulator is intended to provide an environment for investigating improved communication system designs and it is hoped that the results that are obtained from such

telecommunication simulators will help satisfy the ever increasing demands of the telecommunications industry.

It should be noted that the research being conducted at NMSU on TCM is being directed by Dr. Frank Carden. The development of the simulator was conducted by the author to assist Dr. Carden in the continuing investigation of TCM.

## **INTRODUCTION**

Computer simulation of communication systems has been expanding with the corresponding improvements made in digital system technology. It is becoming increasingly easier to find computers in the market place that can handle the large quantities of data that are generated in many communication simulators. Today's computers can offer greater through-put at faster rates and at lower costs than ever before. The ease with which simulations can now be devised, executed, and evaluated makes this a welcome trend to the telecommunications industry.

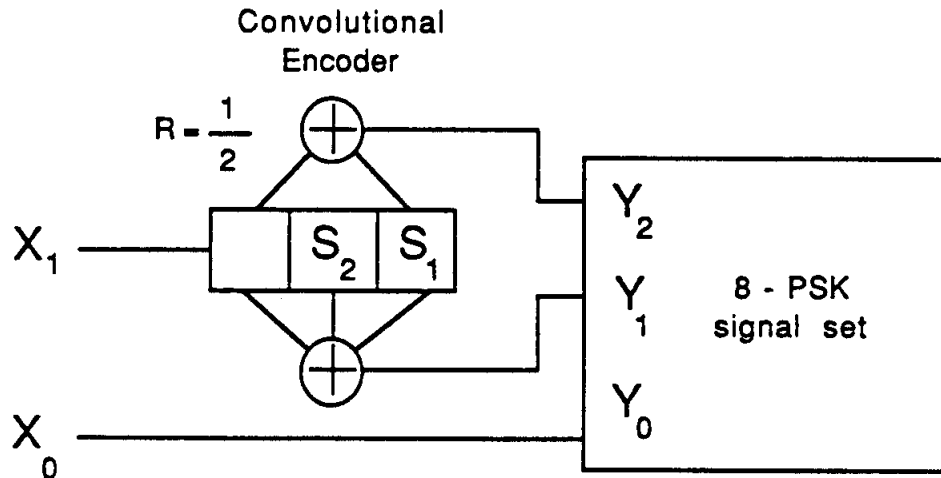
These improvements in digital system technology have even crossed the bounds into the personal computer (PC) market. The PC now has the capability to handle large quantities of simulated communication signals. Simulations of hundreds of thousands and even millions of communication events are now possible with a PC without exceeding acceptable run-times. This paper documents such a simulation.

The communication system that was simulated is 4-state trellis-coded 8-phase-shift-key (PSK) modulation. Trellis coded modulation (TCM) is a relatively new communication scheme that combines the traditionally separate areas of coding and modulation into one system. It combines convolutional encoding with signal set expansion and restrictive signal constellation mapping to achieve substantial coding gains and provide forward error correction. State diagrams that resemble trellises can be used in the decoding process.

For the simulator the TCM scheme was applied to 4-PSK modulation of two data bit words. With TCM the first data bit is sent through a one-half code-rate convolutional encoder and the second is given directly to the modulator. The resulting three bit word is mapped into an 8-PSK constellation thus demonstrating signal set expansion. The TCM block diagram for 8-PSK is shown in figure 1.

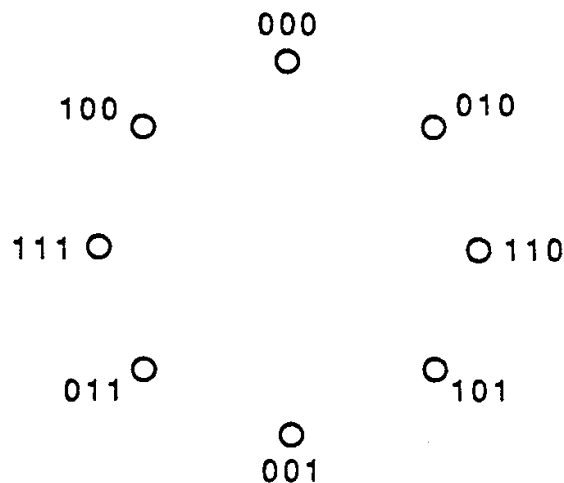
The second data bit from the two bit word is used to identify four nearest neighbors in the 8-PSK constellation. The first data bit from the two bit word which has been converted into two convolutionally encoded bits is used to select one of the four nearest neighbors in the 8-PSK constellation. Because of the nature of convolutional encoders only certain sequences of 8-PSK signals can be selected. Further the mapping of signals is such that the

four nearest neighbors in the 8-PSK mapping that are selected will be opposite from the other four phasors in the constellation only with respect to the least significant bit of the 8-PSK three bit word. An 8-PSK mapping for the TCM system in figure 1 is shown in figure 2.



**Figure 1. TCM block diagram.**

Comparing any four nearest neighbors in figure 2 with the other four constellation phasors it is possible to see that they are opposite only with respect to the least significant bit. This demonstrates the redundancy of the signal set expansion. Redundancy combined with convolutional encoding leads to maximum Euclidean distance in signal constellation sequences and a corresponding coding gain. Gottfried Ungerboeck has shown [1] that 4-PSK expansion to four-state trellis-coded 8-PSK will lead to an asymptotic 3dB coding gain as shown in figure 3.



**Figure 2. Signal constellation for 8-PSK TCM.**

The demodulation of TCM involves trellis decoding using the Viterbi algorithm. In convolutional decoding binary distances are mapped onto the trellis state model branches and cumulative Hamming distances are determined to decide which state transition path represents the maximum-likelihood transmitted path. For TCM the actual Euclidean space distances between the received 8-PSK phasor and its four nearest neighbors are mapped onto the trellis branches. The cumulative distances are compared as in convolutional decoding to determine a maximum-likelihood transmitted sequence determined by the shortest path. A trellis is shown in Appendix 1.

### Program Development

The simulation was written in Turbo Pascal on a Macintosh SE. In its present form it consists of two programs. The first runs the simulation and provides an output file containing the identification numbers of the communication symbols that were decoded erroneously. The second program counts the number of erroneously decoded symbols and reports it to the user. Then for a given signal-to-noise ratio (SNR) that was used to generate the channel noise the bit-error-rate (BER) is calculated and reported to the user. Changing the SNR to several different values and repeating the simulation it was possible to generate a curve similar to that of figure 3. This was considered to be the objective of the simulator. Since Pascal is a highly structured programming language it was easy to divide up the different components of the simulation into different programming procedures. This turned out to be a necessary approach since the simulator had to demonstrate performance capability each time a new section of code was constructed.

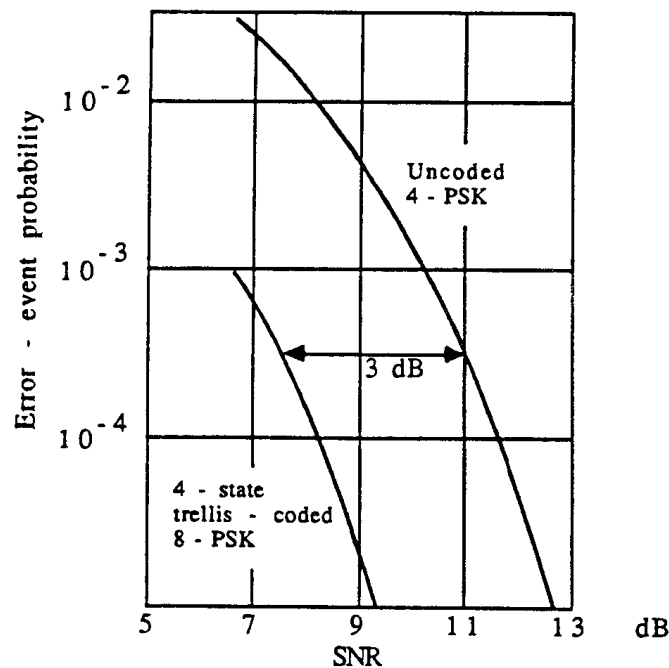


Figure 3. Errors vs. SNR for 4-state trellis-coded 8PSK [2].

Since the program currently exceeds 15 pages of code it is not included in its entirety with this report. However a flowchart is included and can be found in appendix 2. The basic steps of creating a communications simulator are described below. First the data to be transmitted is generated. Second white Gaussian noise (WGN) is added to the signals thus simulating transmission. Third the signals (plus noise) are demodulated and decoded. Fourth the decoded data is compared with the generated data and the number of errors determined. This final step allows a symbol error or error-event probability to be determined thus characterizing the system being simulated.

There are two techniques for TCM simulation that were investigated to determine how best to create the program. Execution time and ease of implementation were the concerns in this investigation. The techniques apply mainly to how best to simulate a Viterbi decoder. This decision would set the format for how the entire simulator design would proceed so it had to be decided upon first. The two techniques are the register exchange method and the trace-back method [3]. According to the Viterbi algorithm for four-state trellis decoding the first technique would require that each of the four surviving information paths into the four states at any point along the trellis be stored in a matrix. Then moving to the next point in the trellis the new stored surviving paths would be a function of the last four paths and the transition branch between them.

“Each time a new branch is processed by the computer, the registers are interchanged corresponding to which sequences survived the comparison, a new symbol is added at one end of each register, and the oldest symbol in each register is delivered to the output decision device” [4].

From a software simulation standpoint this means that at each new point in the trellis the surviving path matrix has to be recalculated and rewritten to a memory device. This is a very time consuming process for a serial computer. Further it is difficult to implement without the help of matrix mathematics intensive software.

The trace-back method is easier to simulate and it is faster than the register sequence technique in a serial computer environment. Trace-back involves retaining the cumulative distances into each state or node. After a certain number of signals are received trace-back begins. This is done by looking at the cumulative distances that represent the possible transmitted paths into any state and deciding what the maximum-likelihood path back to the last state is. This process determines the perceived transmitted symbol that resulted in departing the last state and arriving in the current state. By this technique the entire simulated data-stream can be partitioned into blocks and decoded in the same manner. It should be understood that in a high speed real communication environment parallel processing would be a necessity and would facilitate using the register exchange method in hardware implementation.

Using the trace-back technique it was decided to use data blocks of 25 symbols. A loop was used to multiply the runs of 25 symbols and achieve runs of 250000 symbols. This was a sufficient number of symbols to determine the symbol error probability. Currently the computer takes approximately thirteen hours to process 250000 symbols.

Data generation was achieved by using the random uniform distribution algorithm resident on the motherboard of the Macintosh SE. There are four possible paths leaving a node in the trellis as can be seen in appendix 1. A random number between 0 and 4 was generated to determine which path to take out of a node. The resulting choice provided both a transmitted symbol and a destination node for the next transmission event. This destination node then provided the restrictions for the next possible four paths. The data generator generates 21 data symbols starting at node one and time zero which would be the upper left hand corner of the trellis. Then data samples 22 through 25 are added on to drive the transmitted path back to zero. This is done to simulate an end-of-transmission condition assuming a return-to-zero data format. Listed below is a programming sample demonstrating the data generation process.

```

case node of
1 :
begin
    if (datasample < 1.0) then
    begin
        realx[n] := 0.0;
        realy[n] := 1.0;
        x[n] := realx[n] + noisex[n];
        y[n] := realy[n] + noisey[n];
        node := 1;
    end;
    if ((datasample>=1.0) and (datasample<2.0)) then
    begin
        realx[n] := 0.0;
        realy[n] := -1.0;
        x[n] := realx[n] + noisex[n];
        y[n] := realy[n] + noisey[n];
        node := 1;
    end;
    if ((datasample>=2.0) and (datasample<3.0)) then
    begin
        realx[n] := 1.0;
        realy[n] := 0.0;
        x[n] := realx[n] + noisex[n];
        y[n] := realy[n] + noisey[n];
        node := 2;
    end;
    if ((datasample>=3.0) and (datasample<4.0)) then
    begin
        realx[n] := -1.0;
        realy[n] := 0.0;
        x[n] := realx[n] + noisex[n];
        y[n] := realy[n] + noisey[n];
        node := 2;
    end;
end;
end;

```

The first two statements check the variable node to determine the current position in the trellis. Then the variable datasample is analyzed to decide which node to go to next and which path to take there. The transmitted values for the phasor coordinates are assigned (realx and realy) and noise is added to them. Also node is reset for the next symbol generated.

The noise generator presented special problems in its conception and required normal distribution generators to create white gaussian noise. Since the random distribution resident internal to the Macintosh is a uniform distribution a conversion routine had to be used to generate two normally distributed random variables which could be added to the generated data. Once again speed of execution played a part in the decision on which convertor to use. Equally important to speed are accuracy and distribution limits.

The first convertor investigated was a simple application of the central limit theorem. Simply stated for this application: the sum of any number of similarly distributed random variables results in a normal distribution. This attempt yielded poor results. The sum needs to be large to get a reasonable distribution and this took excessive time. Sums of up to 16 samples were attempted but none met with acceptable results. The number of errors generated was too excessive for the given signal to noise ratio. This may be attributed to the fact that because the uniform distribution used in the central limit theorem is a bounded distribution so is the approximated normal distribution. This appeared to have a tendency to raise the tails ( $x > 2\sigma$ ) of the distribution and could account for excessive errors.

The second convertor that was investigated is one that is popular with many software simulators and games. It jointly generates two random variables that are normally distributed from two uniform distributions. The resulting distributions have a mean of zero and a variance of one. Further this method has no mathematical bounds of any consequence provided the uniform distribution is prepared correctly. It is a time consuming method but is still considerably faster than the central limit theorem approach of summing many uniform distributions. The dual algorithm is shown below where R1 and R2 are uniform distribution samples and N1 and N2 are normal distribution samples.

$$N_1 = \sqrt{-2 * \text{LN}(1-R_1)} \text{ COS}(2\pi R_2)$$

$$N_2 = \sqrt{-2 * \text{LN}(1-R_1)} \text{ SIN}(2\pi R_2)$$

As can be seen from the root argument it is necessary to keep R1 less than one. Both distributions as generated by the computer conformed to the algorithm restrictions by their boundary conditions of being greater than or equal to zero and less than one. After the normal distributions were generated the only thing necessary to do was to multiply them by a constant representing the standard deviation for the noise in an 8-PSK modulated system

with a given SNR. It is assumed that the symbol power  $E_s$  is normalized and equal to one. For the example below assume a SNR of 9.0dB.

$$10 \text{ LOG } \frac{E_s}{N_0} = 9.0$$

$$\frac{E_s}{N_0} = 7.943282$$

$$N_0 = 0.125892$$

$$\sigma_n^2 = \frac{N_0}{2} = 0.062946$$

$$\sigma_v = 0.250891$$

The normal distribution is then multiplied by this standard deviation to arrive at a normal distribution with a mean and variance that conform to the noise model for the 8-PSK modulation scheme. As was mentioned above these noise values are then added to the transmitted signals and the resulting sums passed to the receiver.

The first part of the receiver is the four nearest neighbors algorithm. This part of the simulator decides which four phasors are closest to the received noisy phasor. The resulting four phasors indicate which member of each of the eight pairs of redundant paths the trellis should use to calculate the Eculidean distance. A sample of this algorithm is shown below.

```

if (((x[n]<0.0) and (y[n]<=0.0))) then
begin
  if (abs(x[n])>abs(y[n])) then
  begin

{4 nearest neighbors are : 135,180,225,270}

      x00[n]:=0.0;          y00[n]:=-1;
      x01[n]:=-0.7071;     y01[n]:=-0.7071;
      x10[n]:=-0.7071;     y10[n]:=0.7071;
      x11[n]:=-1.0;        y11[n]:=0.0;
  end;
  if (abs(x[n])<=abs(y[n])) then
  begin

{4 nearest neighbors are : 180,225,270,315}

      x00[n]:=0.0          ; y00[n]:=-1;
      x01[n]:=-0.7071    ; y01[n]:=-0.7071;

```

```

                                x10[n]:=0.7071    ; y10[n]:=-0.7071;
                                x11[n]:=-1.0      ; y11[n]:=0.0;
                                end;
                                end;
                                end;

```

The section shown above represents the 4 nearest neighbors determination for two 45 degree sectors in the Cartesian plane. These particular two represent the third quadrant. Once a received phasor is determined to be located in a certain sector it is assigned four possible transmitted phasors. At this point the redundancies in the trellis are removed and the four sets of coordinates are used to calculate the distance between themselves and the actual received coordinates. This process is shown below.

```

tx[n,1,1]:= x00[n]    ;ty[n,1,1]:= y00[n]    ;
tx[n,1,2]:= x11[n]    ;ty[n,1,2]:= y11[n]    ;
tx[n,2,1]:= x11[n]    ;ty[n,2,1]:= y11[n]    ;
tx[n,2,2]:= x00[n]    ;ty[n,2,2]:= y00[n]    ;
tx[n,3,1]:= x01[n]    ;ty[n,3,1]:= y01[n]    ;
tx[n,3,2]:= x10[n]    ;ty[n,3,2]:= y10[n]    ;
tx[n,4,1]:= x10[n]    ;ty[n,4,1]:= y10[n]    ;
tx[n,4,2]:= x01[n]    ;ty[n,4,2]:= y01[n]    ;

{determine the distance between the trellis coordinates and}
{the received coordinates}

for a := 1 to 4 do
begin
for b := 1 to 2 do
begin
d[n,a,b] := sqrt(sqr(tx[n,a,b]-x[n])+sqr(ty[n,a,b]-
y[n]));
end;
end;
end;

```

The variables tx and ty are used to store the eight trellis branch phasors that remain after removing the redundancies and are assigned according to the four nearest neighbors for that state n. The variable d contains the distances which are summed and examined according to the Viterbi algorithm.

Once the decoding process has started and the four nearest neighbors have been determined for all 25 symbols there remains two paths arriving at any node. According to the Viterbi algorithm the one with the smallest cumulative distance is determined to be the survivor and is chosen to move on through the trellis. This is relatively simple to implement in the simulator with several storage matrices. The trick comes in getting the process started. Since the paths always start at node one for a given block of 25 symbols and keeping the possible exit paths from any node in mind it is possible to see that for the first event nodes 2, 3, and 4 will have no survivor to pass on to the next state. Likewise for the second event nodes 3 and 4 will not have any outgoing survivors. The result is that these first two events must be evaluated differently from the rest of block of symbols. They

must be limited as to the survivor determinations that can be made. The other 23 events can be analyzed in the same manner utilizing a simple comparison loop. The survivor determination process for the first two events is shown below.

```
c[1,1] := d[1,1,1];
c[1,2] := d[1,2,1];
c[1,3] := 0.0;           {assigned only to fill matrix}
c[1,4] := 0.0;           {assigned only to fill matrix}
c[2,1] := c[1,1] + d[2,1,1];
c[2,2] := c[1,1] + d[2,2,1];
c[2,3] := c[1,2] + d[2,3,1];
c[2,4] := c[1,2] + d[2,4,1];
```

The only critical point that can occur in the survivor determination is when the two paths into the same node have the same cumulative distances. Then ‘a coin is flipped’ to determine which one to pick as the survivor. Actually a random number between one and two is used to make the decision. It should be noted that the choice must be retained because it may be needed during trace-back.

Once the survivors have all been determined trace-back begins. Starting at event number 25 the four surviving paths are compared and a winner is declared. The program will not execute trace-back if there is not a clear winner after 25 symbols. The path of the winner is then traced back to event number 24. From event 24 down to event 3 the process is the same. The winner is traced back one event at a time always looking at the results from the survivor determination to decide where to go. Each time a destination determination is made the trace-back technique outputs the assumed transmitted symbol. The symbol represents the trellis branch taken by the trace-back technique during its propagation backwards along the maximum likelihood path. Any time two paths coming into a node have the same cumulative distance during trace-back the simulator looks at the decision made during survivor determination to decide where to go. At event three the trace-back technique terminates. Events one and two are handled differently because of the unique situation that occurs with regard to the lack of a full compliment of survivors. They are merely assigned there assumed transmitted paths based on where the maximum-likelihood path is when it has worked its way down to the third event.

Now that the assumed true path has been determined it must be compared with the correct transmitted 25 symbols and any resulting errors that have occurred must be written to an output file. The number of the error which is between one and 250000 is what is actually written to the file. The Cartesian coordinates of an assumed transmitted phasor are compared with the actual transmitted phasor’s Cartesian coordinates and any discrepancies are considered to constitute an error.

To achieve runs of 250000 symbols the process of analyzing 25 symbols at a time must be repeated 10000 times. Once this is completed the output file is examined. The number of errors is counted and the corresponding error event probability is determined. This is simply a ratio of the number of errors that occurred to the 250000 symbols transmitted.

## **SIMULATOR RESULTS AND CONCLUSIONS**

After the a run of 250000 samples returns an error event probability the simulator is reset with a different standard deviation for noise corresponding to a different SNR. These data points are then plotted on a graph as in figure 3. The tabulated results and the plot from the simulator are in appendix three. Also included in the plot is the simulation that was conducted by Ungerboeck in his article of references [1] and [2]. Although he gave no details as to how the simulation was carried out it is interesting to compare the two. The graph demonstrates that the simulator design is working. It also demonstrates that there is room for improvement. The deviation from the theoretical 4-state trellis-coded 8-PSK curve ranges from approximately 1.5 dB to around 2.5 dB Several areas are going to be investigated to determine how best to improve the simulator.

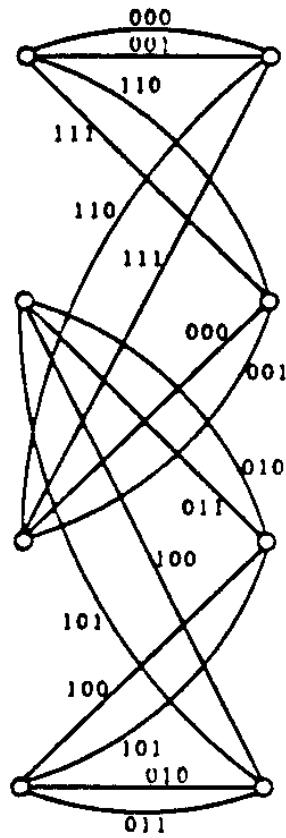
The highest priority is with the random noise generator. It is believed that there is still room for accuracy improvement. The uniform noise generator performs many important tasks in this simulator and the algorithm used by the Macintosh SE may need to be altered to improve its performance. The simulator is going to be up loaded to the campus main frame computer at New Mexico State University for comparison runs there and it is hoped that this too will provide some information as how to improve the simulators performance.

Speed is the second concern. Source code refinements are under consideration at this time. The matrix manipulations are of particular concern here. APL an older programming language that has Strong matrix mathematics abilities is being investigated for possible simulator implementation. Long range plans for the simulator include applying TCM to other modulation schemes. Quadrature amplitude modulation (QAM) and 8-PSK are two of the candidates.

## APPENDIX 1

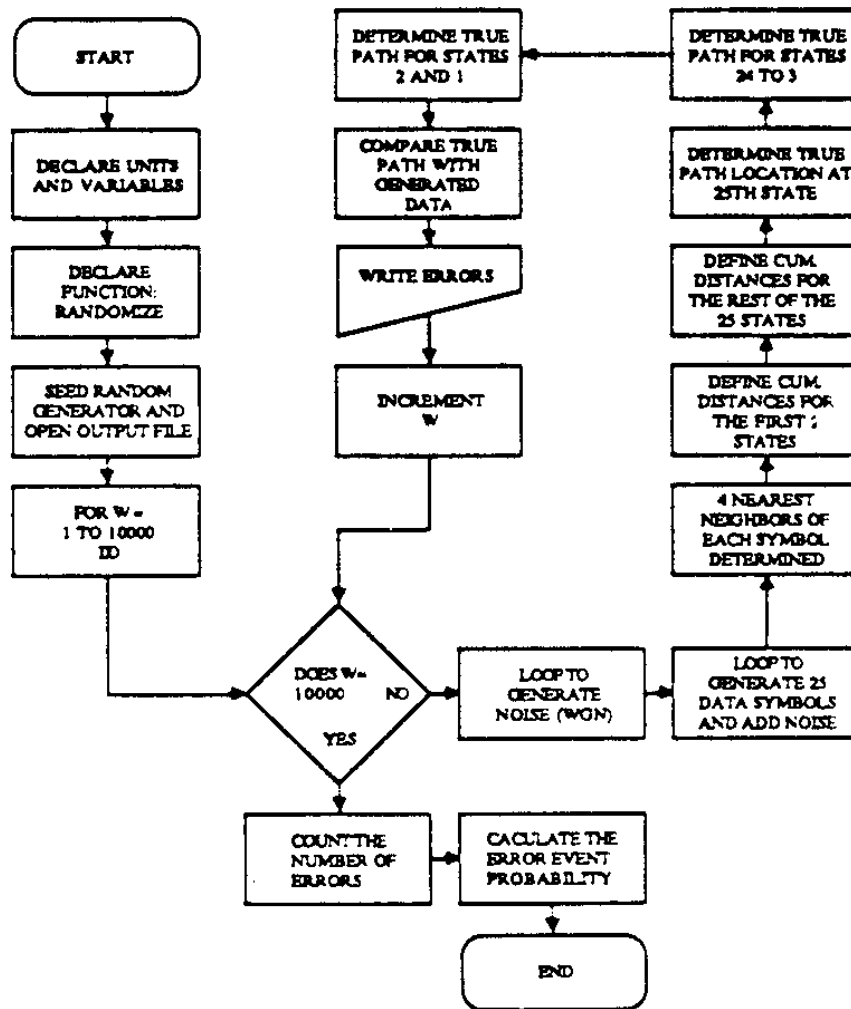
The trellis for TCM.

Note that the actual Eclidean space distances are not mapped onto the branches. This was done for clarity. The symbols representing the 8-PSK signals as shown in figure 2 are included instead. These represent the reference phasors from which the received phasor is subtracted. Each difference which represents the distance between received signal and possible transmitted signals is then applied to the Viterbi algorithm.



## APPENDIX 2

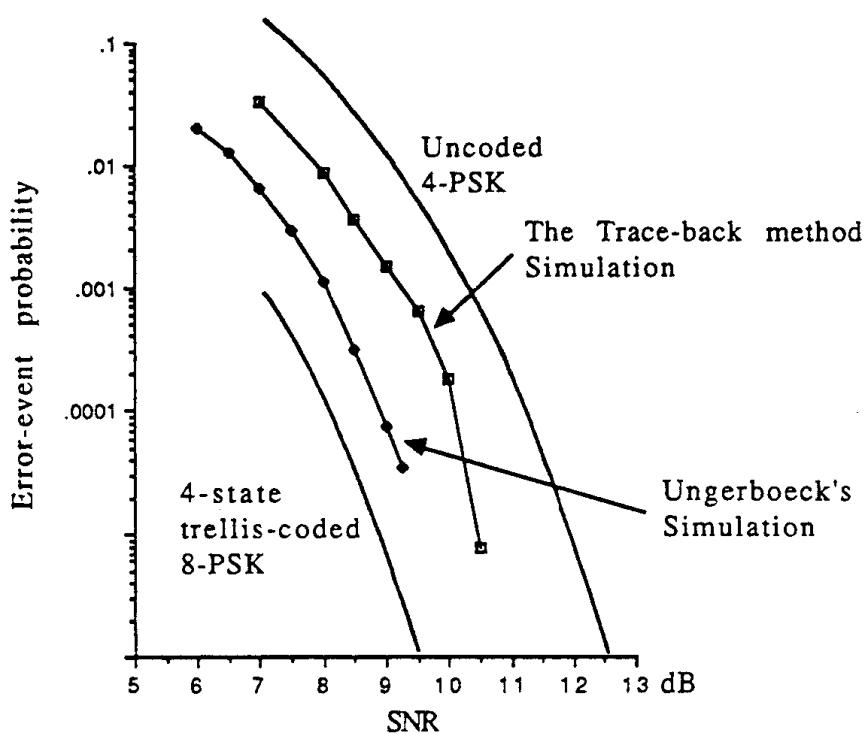
### Simulator Flowchart



## APPENDIX 3

### The Simulator Results

Signal to Noise Ratio	Standard Deviation	Number of Errors	Error Event Probability
7.00	0.315853	8447	0.033788
8.00	0.281504	2226	0.008904
8.50	0.265757	908	0.003632
9.00	0.250891	377	0.001508
9.50	0.236856	160	0.000640
10.00	0.223607	46	0.000184
10.50	0.211098	2	0.000008



## REFERENCES

- [1] Trellis-Coded Modulation with Redundant Signal Sets, Part 1: Introduction, Gottfried Ungerboeck, IEEE Communications Magazine, Institute of Electrical and Electronics Engineers, New York, Vol. 25, No.2, 1987.
- [2] Trellis-Coded Modulation with Redundant Signal Sets, Part 1: Introduction, Gottfried Ungerboeck, IEEE Communications Magazine, Institute of Electrical and Electronics Engineers, New York, Vol. 25, No.2, 1987.
- [3] Error-Correction Coding for Digital Communications, George C. Clark, Jr. and J. Bibb Cain, Plenum Press, New York and London, 1981, pg. 261.
- [4] Error-Correction Coding for Digital Communications, George C. Clark, Jr. and J. Bibb Cain, Plenum Press, New York and London, 1981, pg. 261.

## BIOGRAPHY

Brian T. Kopp was born in Greenwich, Connecticut, on August 25, 1964. He is a senior in Electrical Engineering at New Mexico State University. He will begin his final semester for a B.S.E.E. in the fall of 1988.

Currently he is doing research with Dr. Frank Carden in the area of Trellis-Coded Modulation, a proposed communications scheme for the NASA space station Flight Telerobotics Servicer.

Brian is a member of Eta Kappa Nu and the IEEE Aerospace and Electronics Systems Society.