Study into Synchronization Scheme for Improved Spectrum Efficiency for FDMA/TDMA Transmission in Mobile and Mobile Satellite Environments[1]

W F McGee

Contract U6800-9-0526

06/22/99 1:21 PM

Abstract

The phase synchronization of a frequency division multiplexing system using spectrally-efficient overlapping carriers is evaluated by computer simulation. The system is implemented with tree-structured filter banks using linear phase filters, and provides bandwidth-on-demand by allowing data to be introduced at various levels of the tree. Synchronization is of the pilot-symbol assisted method which is useful in Rician-channel fading environments with Doppler shifts typical of the mobile and mobile satellite environment. The pilot symbols are inserted using a novel method, suggested by John Lodge, by creating a dead-zone in an intermediate data stream in which the pilot sequence is introduced. Phase and time synchronization are effected in the receiver with a filter matched to the transmitted pilot symbol sequence, and the channel phase is determined by interpolating these periodic samples of the channel response. The report discusses the simulation techniques based on a universal tree structure exploiting virtual filters to represent data introduced at higher levels, provides a fundamental theoretical analysis of the synchronization method that agrees with simulation, considers the effects of a variety of Doppler shifts which will allow application guidelines to be deduced, and provides a qualitative explanation for these results. The filters are based on a newly-derived and unique set of prototype linear phase root-Nyquist filters with equiripple stopband losses. Fundamental properties of 2-channel prototype filter banks such as bandwidth, loss and use in VSB configurations are reviewed. A simplified receiver is analyzed.

ncial support under Industry Canada's Spectrum Research Funding is gratefully acknowledged.

Please make the following changes to the report:

p. 27 replace "staggered QAM" with "offset QPSK".
p. 55 replace "receive filter" and "channel filter" with "effective phase filter with response sin(N pi f/fs) with N=32"
p. 57 After the sentence "Golay has considered such sequences, " delete all the text up to the sentence beginning "He chooses...".
p. 58 Replace "minimize" with "maximize".

17 Jan 1999

I. Introduction

A. Introduction and Contents of Report

SYSTEM CONFIGURATION

In the considered systems, terrestrial transmission facilities are connected by digital trunks to a satellite ground station. In the ground station, a digital to FDM converter creates a composite FDM signal in which the sidebands for individual channels overlap, creating spectral efficiency. This composite FDM signal is sent to a satellite which relays it to mobile stations, called the downstream direction. In the mobile station a receiver demodulates the FDM signal associated with its carrier. The channel carriers and bandwidths are assigned using a communication link.

In the reverse, upstream, direction, when authorized by the communication link, individual mobile stations transmit FDM signals to a satellite. After amplification in the satellite, this signal is relayed to the ground station. Here the individual FDM signals are recovered and separated and used to form a digital signal suitable for trunking.

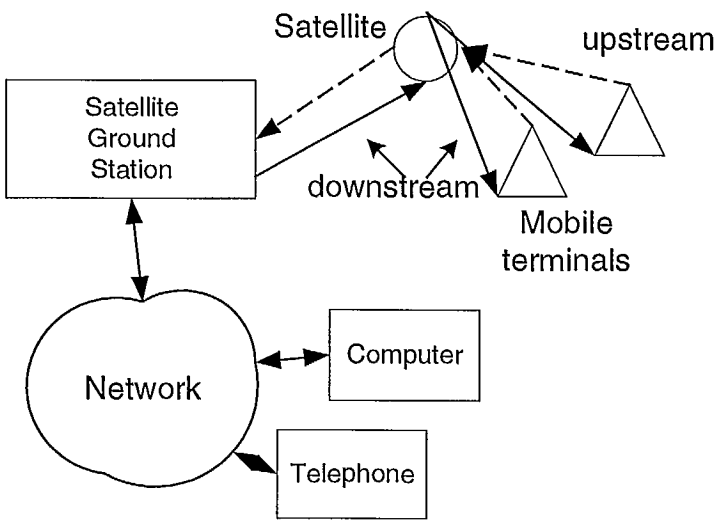


Figure 1 This study is of the downstream direction of satellite to mobile communication. A typical communication application is shown in this figure to illustrate how this fits into an overall network. For example, a computer sends data to the network, which connects to a satellite ground station. The information is processed in the ground station and transmitted through a satellite to mobile terminals. In the upstream direction not considered in this report, and shown with dotted lines, mobile terminals transmit to the satellite which relays this information to the satellite ground station for processing, and transmission to the network.

Such a system is attractive because it requires no on-board satellite processing, permits a simple per channel mobile station transmitter-receiver, and is spectrally efficient. There is ground station processing, but the multicarrier system probably requires backoff of the satellite amplifiers to avoid saturation.

MULTIPLEXING AND MODULATION

There appear to be three approaches to frequency division multiplexing for this application. One approach uses the discrete Fourier transform, and is called OFDM. Because of the desire that the signal processing be done in blocks, guard bands in time are usually utilized in OFDM systems, in the form of repeating data from the beginning at the end of the block, resulting in a 'cyclical prefix'. This introduces a slight performance penalty. Generally speaking, the receiver for OFDM signals demodulates all the channels.

The second approach makes use of non-overlapping spectra. This produces a loss in frequency spectrum efficiency which may be made as close to unity as desired at the expense of longer and longer digital filters to improve the selectivity. Quadrature amplitude modulation may be used.

The third approach, and the one used in this report, makes use of vestigial sideband digital signals which are phased appropriately with respect to the adjacent carrier, and which have sufficiently selective filtering to avoid interference into non-adjacent carriers. This is in contrast to OFDM, in which the filtering is not very frequency selective.

BANDWIDTH ON DEMAND

As communications moves towards packet communications away from fixed bandwidth services, even for voice, but especially for Internet connection, the traffic has different characteristics. These may be characterized by asymmetry if required capacity, with the downstream direction generally requiring greater capacity compared to the upstream. Secondly, the traffic patterns are very bursty, often with very fast file transfers representing pictures or files, interspersed with lower rate services. Multiplexing arrangements such as ALOHA and Ethernet offer system flexibility but at the expense of bandwidth inefficiency, typically about 25 percent. The current approach considers variable bandwidth services in which the bandwidths may be allocated electronically, and efficiently.

The particular approach to bandwidth on demand that is used in this project is to use tree-based filter banks. Other possibilities are feasible, such as time division multiplexing or OFDM, but these must demodulate the entire signal; filter trees allow the possibility of retaining the frequency division approach used until now for most radio services.

One of the problems with variable bandwidth is that the configuration in use must be negotiated, or at least communicated, between the transmitter and the receiver. This is most likely to be done with a fixed communication channel to which a receiver may always tune, independent of the configuration.

The second problem with variable bandwidth arises when we consider the necessity to synchronize the oscillators and timing circuits of the transmitter and receiver. Here again it is necessary to have a scheme that does not depend on the system configuration, since the receiver, at startup, does not know the configuration. This requirement is met by the analyzed approach.

PILOT SYMBOL INSERTION

The approach of this report is based on the use of inserted pilot symbols. According to this procedure, pilot symbols are periodically introduced into a data stream, at an appropriate level in the tree. The particular level in the tree chosen to introduce the pilot symbols would generally correspond to the highest data rate that would be offered to users, this may not be the highest data rate used in transmission. Thus, the pilot symbols could be introduced at an intermediate level in the tree. Once this intermediate rate is chosen, subsequent multiplexing would invariably be done with a polyphase filterbank, since this is more efficient. This project has considered a three-level tree, thereby offering the variable relative data rates of 1, 1/2, 1/4 and 1/8.

The central question of this report is to insert the pilot symbols into a data sequence, perform channel tracking, and recover the data sequence. This is a problem with narrowband channels, since the requisite

filtering causes a spreading of data symbols into adjacent time samples; into the sample positions where we would like to introduce the pilot symbols.

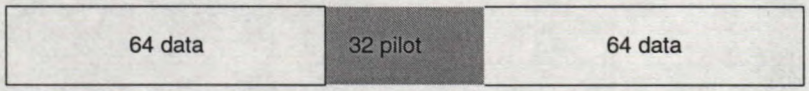


Figure 2 Format of basic data block of 160 symbols. The pilot symbols are in the middle of the block. The blocks are concatenated with no guard symbols.

The approach that is studied is conceptually rather subtle, but has the following explanation. Suppose, for the sake of example, that we were able to insert correction symbols at the input to the tree where there was no data, and corresponding, roughly, to positions that would occur when the pilot symbols should be introduced. Then it should be possible to choose these inserted correction signals to cancel the signals occurring in the pilot symbol positions, so that there would be no signal energy at the positions higher in the tree where we insert the pilot sequences. Then at the receiver, the positions where the pilot sequences are inserted are made equal to zero, and this causes the receiver to generate the correction signals, in which the receiver has no interest, and therefore discards them. This is illustrated in the figure.

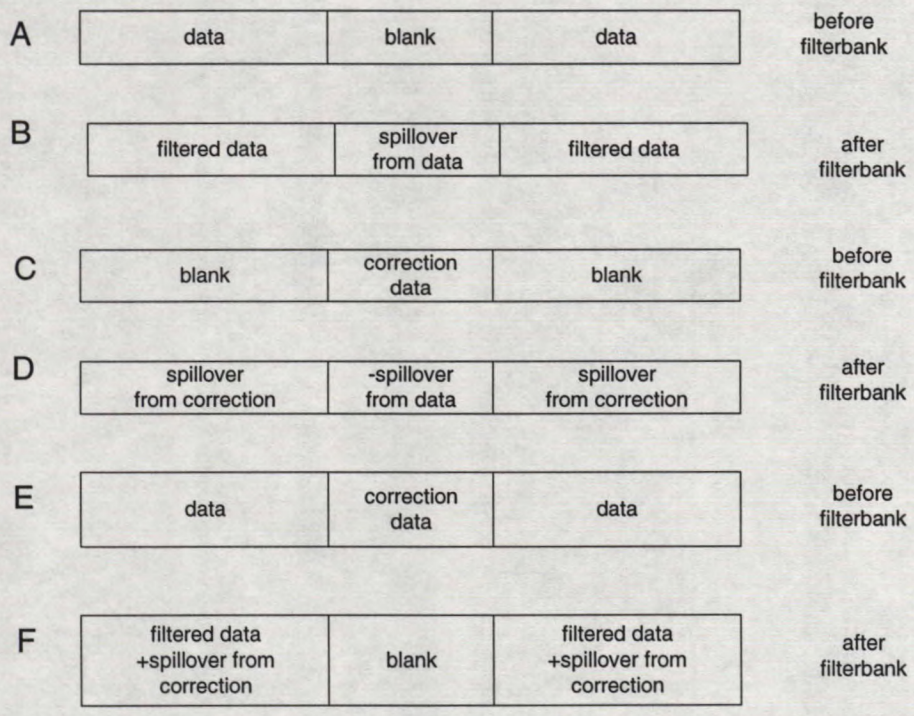


Figure 3 Conceptual view of pilot-symbol insertion at the transmitter. Consider A, where data is inserted at the input and blank (i.e. zero data) is inserted at the input to the filterbank. After filtering, B results, and there is spillover into the position in the middle where we wish to insert the pilot. But this may be cancelled with C, where correction data is inserted such that, after filtering, the negative of the spillover in B results. Then if we add the inputs, as at E, since the filter is linear, the result is F at the output, and the pilot may be inserted without interference. At the receiver, because of the properties of perfect reconstruction filterbanks, if F is presented, E will result at the output, and the data may be recovered, and the correction data is irrelevant.

THE SIMULATION

The purpose of the simulation is to verify that the proposed synchronization method is effective for communication over a Rician channel with typical Doppler shifts, and with typical additive Gaussian noise. The simulation is of the following experiment. Three blocks are transmitted in sequence, containing random data and three pilot sequences. The channel is estimated for each of the three pilot sequences. Then a channel estimate for all the data positions in made by interpolating these three channel estimates, and the received signal is multiplied by the conjugate of the channel estimate. The resulting signal is used to recover the data.

CONTENTS OF THE REPORT

We first consider the hardware building blocks of the system in detail. The primary component is a two-channel elementary trees with root-Nyquist half-band filters. A modification of these trees to form VSB signals is presented. Then the use of 2-channel trees to build a 3-level, 8-channel tree is explained, and the requirements on the filter bandwidths summarized. Then the use of these trees to allow bandwidth on demand is outlined, and the numbering scheme used in the simulation to keep track of the 26 different configurations.

The next two portions of the introductory portions of the report summarize previous work on phase synchronization for Rician channels, and on the use of tree-based filterbanks for communication, including previous CRC work.

The second part of the report discusses the simulation, starting from the block diagram, and detailing the procedures used in the simulation. These include the filter banks, the calculation of the correction data, and the synchronization approach using offset QPSK signaling,

The third part of the report outlines some results of the simulation. These include curves of bit error probability as a function of signal-to-noise ratio, Doppler shift, Rician factor, and system configuration. The extra power needed for synchronization has been analyzed in detail, and the confirming simulation results are presented. The variance of the phase has been measured, and its agreement with theory is reported. Finally, some experiments on the concept of the simplified receiver are given.

Appendices are used to present more involved results. The pitfalls of Rician channel simulation are given, and our approach to yielding reasonable simulation results is presented. The theoretical analysis of Rician channels is given; these are used to benchmark the simulation results. The detailed analysis of the synchronization scheme is explained. A crude analysis of the effects of Doppler shift on performance is given. A short discussion on good sequences for pilot-symbols is presented, although the simulation has used an m-sequence derived sequence specified in the contract. The appendices conclude with the details of the filters used in the simulation, and their frequency response.

The report concludes with the annotated MATLAB listing of the program RSIM that has been prepared for this study.

B. Two-Channel Elementary Trees

The transmitter and receiver use 2-channel filter pairs. Here is a review of their operation. The mathematical basis for these pairs is that, given a scalar polynomial H(z), H(z)H(-z) has no odd-powered coefficients. Thus, we have no crosstalk between adjacent channels in the following configuration, in which input samples are introduced at even sample times, and output samples are taken at odd sample times.

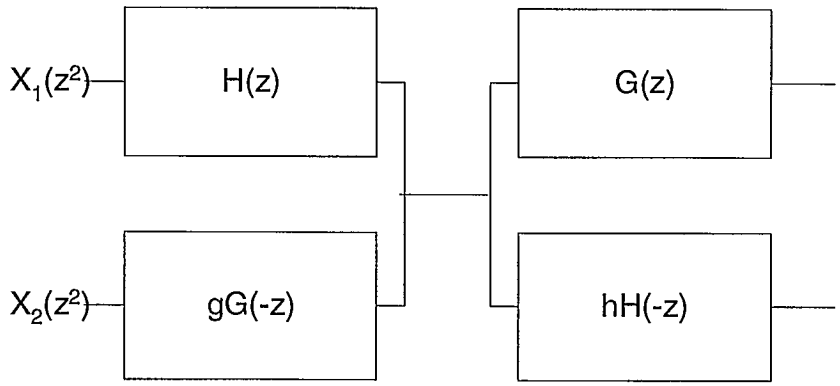


Figure 4 The basic structure of a 2-channel filterbank. There is no crosstalk between the adjacent channels at the output if samples are taken as odd sample times. Here h and g are real numbers, usually +1 or –1.

The above description applies to any polynomial pairs, but to control intersymbol interference, the two filters H and G must be chosen properly. Generally speaking, the pair H(z)G(z) must have one peak at an odd sample time, the other odd-sample-time samples should be small, and there is no restriction on the even samples. If the non-peak odd-time samples vanish we say the resulting system exhibits *perfect reconstruction*; intersymbol interference vanishes.

There are some simple choices. The simplest is the time-division multiplexer, which alternates the samples. These have $H(z)=1$, $G(z)=z^{-1}$. There is no filtering action.

For many applications in communications, the filters H and G have the same magnitude frequency response, and are matched to each other. This means that the coefficients of H are those of G in reverse order. H and G are called root-Nyquist half-band filters. The standard procedure to design these pairs is to design a Nyquist filter N(z) with double zeros on the unit circle, with no odd coefficients except one, and with suitable stopband properties, and then to perform a spectral factorization of N(z). This is simple, because the double zeros on the unit circle are known; only the zeros off the unit circle need to be found, and these are singletons. The zeros of N(z) will have symmetry with respect to the unit circle, and these zeros may be chosen in pairs to be assigned to H and G. This the approach taken in our previous publications [17-20].

For a variety of reasons, it is useful to have linear phase filters for H and G; in fact, it is useful to have H=G. These may be designed with small, but not vanishing, intersymbol interference. The problem, basically, is that if we insist that the intersymbol interference vanish, there are no more degrees of freedom left to specify the attenuation, and we end up with $H=1+z^{-K}$, K odd, and these filters are, for our application, useless.

One set of filters has been given by Johnson[7], but these do not have equiripple stopband loss. Techniques have been developed that yield filters with equiripple stop bands. As an example, consider the coefficients of a polynomial H(z) of Table 1, with the response N(z)=H(z)H(z) given by Table 2. The odd coefficients, except for one, are small, as required.

0	-6.8728e-003
1	3.3712e-002
2	-8.0974e-003
3	-1.2497e-001
4	1.3279e-001
5	6.8227e-001
6	6.8227e-001
7	1.3279e-001
8	-1.2497e-001
9	-8.0974e-003
10	3.3712e-002
11	-6.8728e-003

Table 1Coefficients of f1 The filter has symmetrical coefficients.

0	4.7236e-005
1	**-4.6340e-004**
2	1.2478e-003
3	**1.1719e-003**
4	-1.0186e-002
5	**1.5991e-003**
6	5.0092e-002
7	**-6.2971e-005**
8	-1.5328e-001
9	**2.0510e-004**
10	6.1452e-001
11	**1.0000e+000**
12	6.1452e-001
13	**2.0510e-004**
14	-1.5328e-001
15	**-6.2971e-005**
16	5.0092e-002
17	**1.5991e-003**
18	-1.0186e-002
19	**1.1719e-003**
20	1.2478e-003
21	**-4.6340e-004**
22	4.7236e-005

Table 2 Coefficients of the convolution of f1 with itself. Note that all the odd coefficients are small except the eleventh. The even coefficients are not of interest.

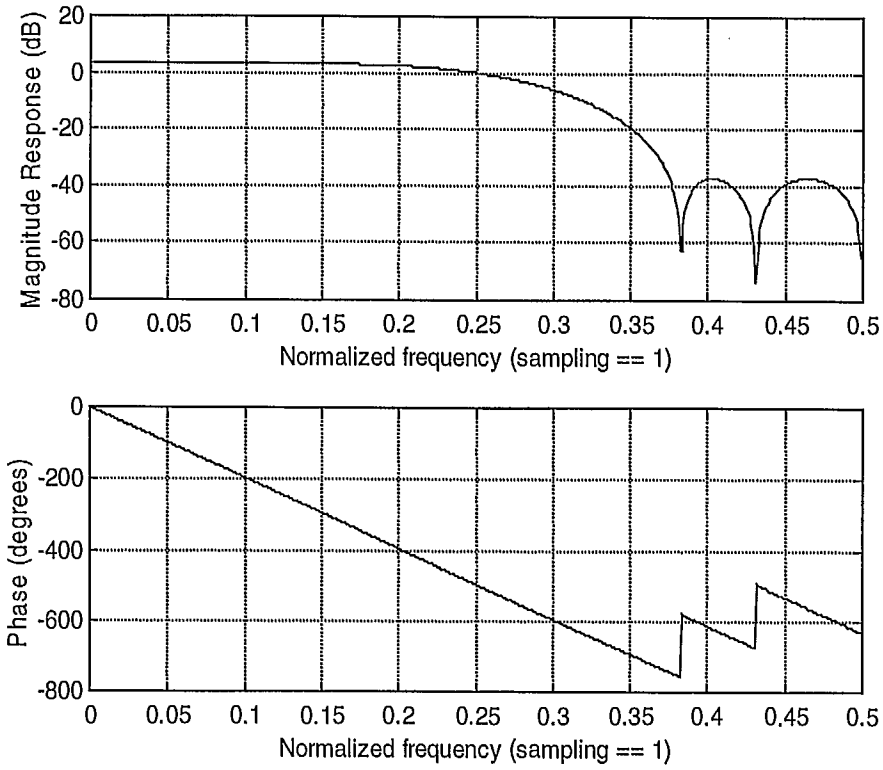


Figure 5 Frequency response of filter f1. There is 40 dB loss in the stopband relative to the loss at 0. These are called 50-percent excess bandwidth filters because the stopband starts at 0.375, and the excess bandwidth is (0.375-0.25)/0.25=0.5. The phase is linear; the delay is constant with frequency.

In our application we restrict our interest to linear phase, lowpass, half-band root-Nyquist filters; thus H(z)=G(z). The filter H(-z) is a highpass filter.

The usual description of these filters makes the interpolation and decimation process explicit. The circle with an up arrow is a zero interpolator, creating a new sequence at double the rate with zeros at the odd time positions. The decimator takes an input and outputs every second sample. There is a phase ambiguity in the decimator, which causes much confusion.

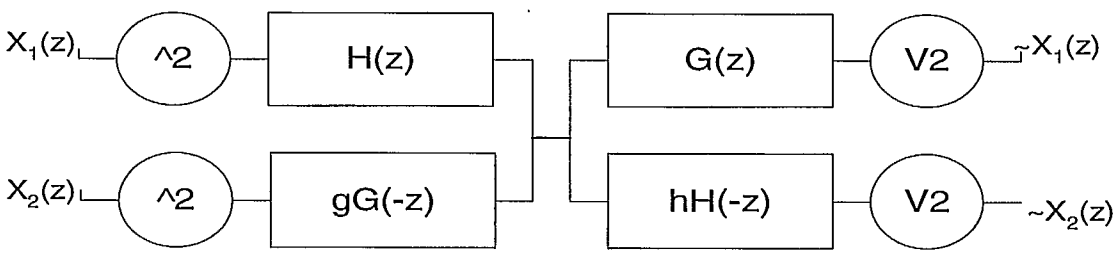


Figure 6. The usual representation for the 2-channel elementary tree.

It is useful to recall that these signal processing pairs appear to have had their origins for speech processing in the arrangement of the following figure, in which the analysis pairs and the synthesis pairs are interchanged.

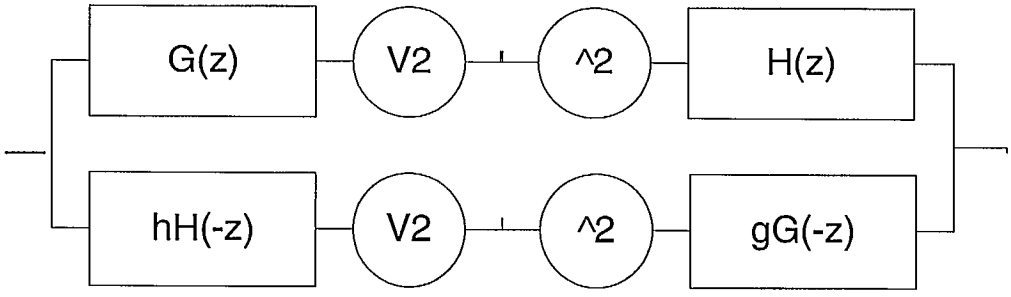


Figure 7. The "signal processing" arrangement of the filter bank. An input signal is analyzed by the two filters H(-z) and G(z), decimated at odd sample times, and these samples are interpolated with the filters H(z) and G(-z) to produce an output that is a replica of the input. With perfect reconstruction pairs, the output is the input.

The argument is as follows. The process of decimation and interpolation is equivalent to taking a sequence, and retaining only its odd samples. For a sequence X(z), this is (X(z)-X(-z))/2. Therefore, the output of the device of Figure 3 is

$$\frac{1}{2}\{[G(z)X(z)-X(-z)G(-z)]H(z)+hg[H(-z)X(z)-X(-z)H(z)]G(-z)\} \quad (1)$$

$$=\frac{1}{2}[G(z)H(z)+hgH(-z)G(-z)]X(z)$$

$$-\frac{1}{2}[G(-z)H(z)+hgH(z)G(-z)]X(-z)$$

The second term is undesirable aliasing of the input signal appearing at the output, and may be eliminated by setting hg=-1, no matter what the polynomials H and G are. The first term represents amplitude and frequency distortion of the input signal, and should be made small. In the ideal case we have

$$G(z)H(z)-G(-z)H(-z)=2z^{-K} \quad (2)$$

This equation states that there is no distortion if the odd part of G(z)H(z) has one non-vanishing [odd] coefficient equal to 1. The integer K must be odd. This is simply a restatement of the condition given in the previous section. With quasi-perfect reconstruction, the overall odd-part transfer function may be as close to unity as possible; the deviations are called the reconstruction error. Generally speaking, the same filters work for signal processing and communications, and linear phase polynomials are often desirable in both applications.

C. VSB Elementary Tree

The 2-channel arrangement may also be used for VSB signaling, with some slight modifications. The goal here is to take two real sequences and to combine them to form a complex sequence, and have the frequency spectrums of the initial sequences separated in the complex sequence. That is, if the resultant complex signal is frequency analyzed, the two constituent signals are, ideally, frequency disjoint. One

could form a complex QAM signal by simply taking the real part to be one sequence and the imaginary part the other sequence, but this would mix the spectrums. In fact, for VSB signaling, one of the sequences uses the positive frequencies of the complex sequence, and the other the negative frequencies. This is done by taking a prototype lowpass filter pair H(z) and G(-z) and frequency shifting H to occupy the positive frequencies, and G(-z) to occupy the negative frequencies. Then the two filters are H(-jz) and G(jz). In the linear phase case these are H(-jz) and H(jz).

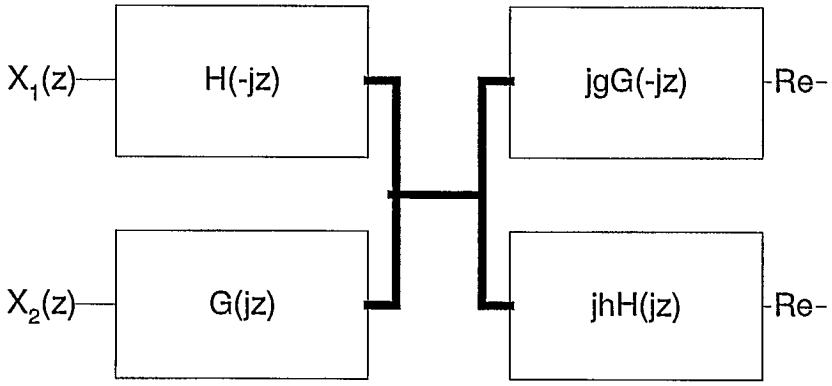


Figure 8. VSB filter pair. The inputs $X_1(z)$ and $X_2(z)$ are real sequences. Dark lines indicate a complex sequence, narrow lines real sequences. There is no interpolation or decimation.

The operation is as follows. Since linearity and time invariance apply, we consider one input pulse, and show that it may be recovered independent of all the other inputs. Consider first a crosstalk path, such as H(jz)jhH(-jz). The polynomial product H(jz)H(-jz), where H has real coefficients, has only real coefficients. Therefore, multiplication of this product by the imaginary constant jh, produces a transfer function such that, if at the output the real part is taken, there is no crosstalk. Within the channel itself, the transfer function is H(-jz)jgG(-jz) where H(z)G(z) is a Nyquist pulse. That is, it has only even order coefficients, and one odd coefficient. But the even coefficients of H(-jz)G(-jz), then, must be real, and if we multiply by jg, an imaginary constant, and take the real part at the output, there is no intersymbol interference from symbols spaced an odd number of samples away, and the Nyquist conditions assures that there is no crosstalk from samples an even number of samples away. Finally, the main pulse is detected because at an odd sample number the product H(-jz)G(-jz) is imaginary, and by multiplication by jg, with g of the appropriate sign, the sample itself is recovered. The resulting signal is a vestigial VSB signal; vestigial because there is some overlap of the signal spectra.

A related approach may be used, called Offset QPSK, [QPSK = quadrature phase shift keying] among many other names. Although conceptually more complicated than VSB, we prefer Offset QPSK for phase recovery. An offset QPSK modulator takes a real sequence and forms a complex sequence whose even coefficients are the same, and whose odd coefficients are the same as the odd coefficients of the original sequence, multiplied by j. An offset QPSK demodulator takes the input sequence and passes the even-numbered coefficients unchanged, and passes the imaginary part of the odd-numbered coefficients unchanged. The term offset QPSK is used because this is sometimes viewed as taking a QAM sequence and offsetting the imaginary parts by half a sample time.

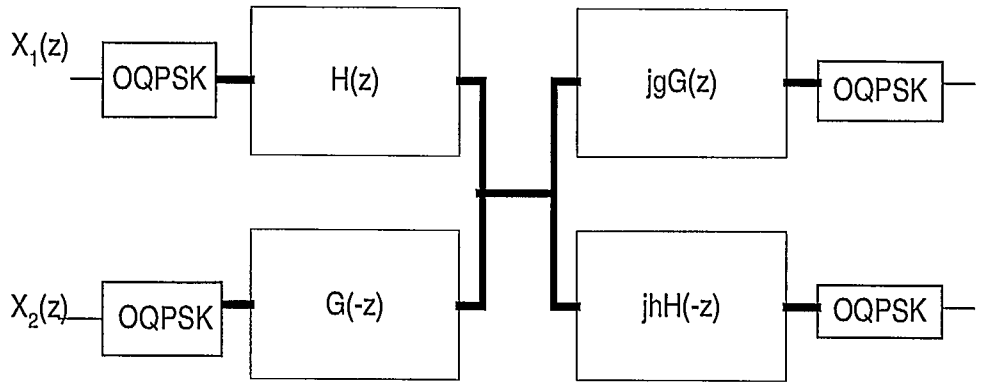


Figure 9. OQPSK used to combine real sequences. The dark lines represent complex signals, the narrow lines represent real signals.

Offset QPSK (OQPSK) operation is as follows. First consider the crosstalk path with transfer function jhH(z)H(-z). This transfer function has no odd coefficients, and the even coefficients are purely imaginary. But the OQPSK demodulator takes the real part of even coefficients, so there is no crosstalk from the even-numbered samples of the input $X_1(z)$. A similar arguments shows that there is no crosstalk from the odd-numbered samples of $X_1(z)$. In the main path H(z)G(z) is a Nyquist pulse and so has vanishing odd coefficients except one. Thus jgG(z)H(z) has vanishing odd coefficients except one, which is assumed, properly normalized to be j, and the even samples are all imaginary. Consider an input sample occurring at time 0, and that the overall delay, which is an odd integer, is 7, for example. Then the input sample is maintained real, appears at sample 7, is multiplied by j, and recovered with the OPSK demodulator, which takes the imaginary part of odd samples. There is no intersymbol interference from even numbered samples by the Nyquist property. Consider an odd numbered sample. It has been made imaginary in the OQPSK modulator, appears at sample 7 imaginary, is multiplied by jg to make it real, and the OQPSK demodulator, which takes the imaginary part at odd samples, ignores this real sample.

Another approach to OQPSK is to consider an alternative modulator that forms X(-jz) from X(z), which is equivalent to a frequency shift of ¼ the sampling frequency. After filtering and combination, the resulting complex signal is down-converted by 1/4 the sampling frequency and real part of the resulting complex signal used to recover the original data.

It should probably be pointed out that these VSB and offset QPSK arrangements rely on the inputs being real. Two-channel filter banks discussed in the previous section may be equally well used for complex sequences, but the resulting frequency spectra result in two sidebands.

D. Tree-Structured Filter Banks : Filter Design

In this section the use of elementary 2-channel blocks to make a tree-structured filter bank is reviewed. The requirements for the filter design are derived.

The elementary 2-channel blocks of section 1 may have their outputs assigned as inputs to another block, and thereby a full binary tree developed. It should be clear that if the appropriate delays in decimation are maintained, the input signals may be recovered at the output with no adjacent channel interference and little intersymbol interference. Thus, the question that remains concerns the frequency response of the filters, and the configurations for the various bandwidth on demand choices.

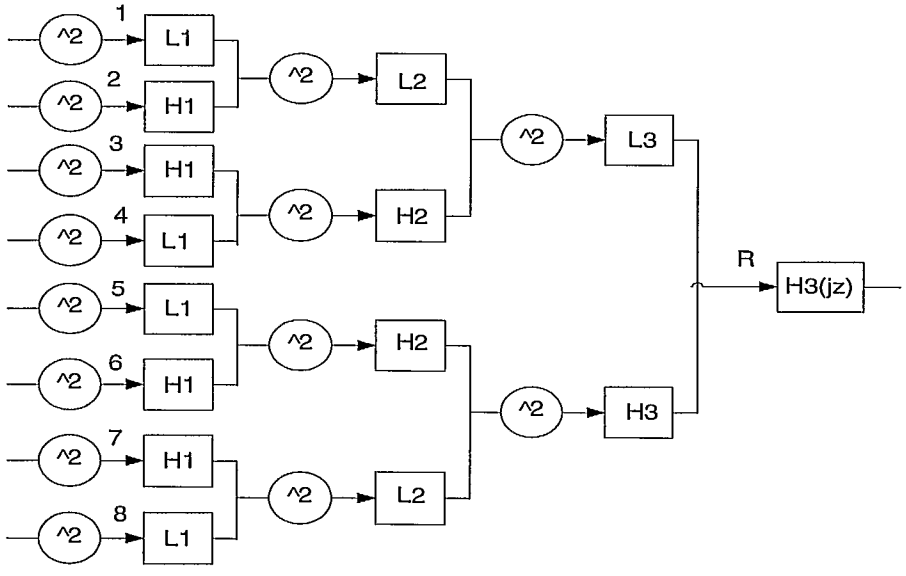


Figure 10. Tree-structured filter bank used to multiplex data signals to a composite output signal. There are three levels to the tree, and the root is connected to a VSB filter.

Frequency Response

In the theory of wavelets, one prototype filter design is used throughout. In fact, the same prototype design is used to design the root-Nyquist filter that is used to bandlimit the digital sequences. But there is no reason to restrict ourselves by this requirement. Indeed, instead of requiring that the pulse shapes be similar, it is far more useful to have the frequency responses of the filters in the transition bands similar. In fact, we show that one filter for each level and suitable demodulation is all that is required for the receiver, greatly simplifying the signal processing required in the receiver, if the prototype filters, in addition, have linear phase. Thus, although it is possible to design one filter for the various levels of a tree, it is far more useful to use a variety of filters, one for each level. In general, a filter handling a higher rate signal is longer because the transition band is half as wide. This means that the amount of processing per input sample is approximately constant at each level and therefore the total amount of processing is proportional to the number of levels. Finally, the VSB filter at the output has the same prototype as the longest prototype filter.

We have shown [17] that if the first level filters have a transition band with an excess bandwidth of 50 percent, at the next level 25 percent, the next 12.5 percent and so on, the system requirements are met. The argument is repeated here; refer to [17] for more details.

In order to understand this argument, recall that the transfer function from the input to the root of the tree consists of a product of transfer functions, the last with argument z, the second last with argument z^2, the third last with argument z^4, the fourth last with argument z^8, etc. The higher and higher powers of z arise from the interpolation.

Consider the transfer function from the point labeled 1 in the Figure 10 to the point labeled R (for root). The transfer function is

$$L_1(z^4)L_2(z^2)L_3(z)$$

and L_1 is a prototype half-band filter with excess bandwidth ratio 1/2, L_2 is a prototype half-band filter with excess bandwidth ratio ¼, L3 is a prototype half-band filter with excess bandwidth ratio 1/8. The excess bandwidth ratio for a root-Nyquist filter is the ratio of the bandwidth not in the stopband to the minimum bandwidth required for data transmission according to the Nyquist criterion, less 1. Thus, with L1, with excess bandwidth ratio 1/2, the stopband starts at ¼+1/4*(1/2)=3/8, the stopband of L2 starts at ¼+1/16, and L3 starts at ¼+1/32.

The frequency response of $L_1(z^2)$ has the frequency axis shrunk by a factor of 2, and repeats. Thus the filter has a stopband from (1/2)(3/8)=3/16 to (1/2)(5/8)=5/16, and two passbands, one centered at dc and one centered at the frequency ½. The purpose of the filter L_2 is to suppress the upper passband, and to do this it must have its stopband start at 5/16, and thus it requires and excess bandwidth ration of (5/16-1/4)/(1/4)=1/8.

The condition that we have been imposing is that stopbands of the nearest filters that are not adjacent should start at the same frequency. We plot the filter responses for the various channels in the accompanying figure.

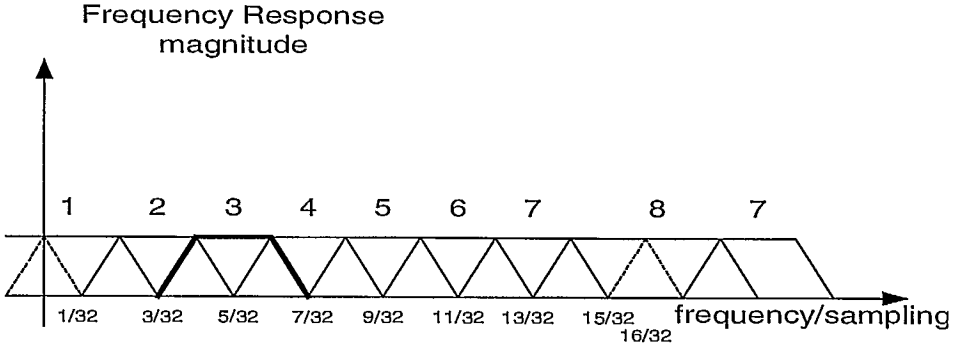


Figure 11 Magnitude frequency response from points indicated with integers in Figure 10 to point R. Filter 1 has a stopband starting at 3/32, filter 2 at 1/32 and 5/32, ..., filter 7 from 11/32 to 15/32 and filter 8 from 13/32 and 19/32. The dotted lines indicate the VSB filters used to reject the positive or negative frequencies. The response of channel 3 has been darkened.

The filter pair $L_1(z^4)$ and $H_1(z^4)$ define the transitions about the frequencies 2/32, 6/32, 10/32 and 14/32. The excess bandwidth ratio is thus 1/32/(1/16)=1/2. The filter pair $L_2(z^2)$ and $H_2(z^2)$ define the transitions

about the frequencies 4/32 and 12/32, and therefore require an excess bandwidth ratio of (1/32)/(4/32)=1/4. The filter pair $H_3(z)$ and $L_3(z)$ define the transition at 8/32=1/4, which has an excess bandwidth of 1/32)/(1/4)=1/8. The VSB filters are also shown. The positive frequency filter has a stopband starting at –1/32 and 17/32, and it is clear from the figure that an excess bandwidth ratio of 1/8 will suffice. Similarly, it is clear that the argument generalizes to a tree with any number of levels.

In this section a description of tree-structured filter banks has reviewed the structure, and outlined the requirements for the filter specification. The requirements for VSB filters have also been derived. The filter bank overall produces a set of bandpass filters which produce pulses which only overlap in spectrum in the adjacent frequency channels. The signals in the adjacent channels are phased so that the crosstalk between adjacent channels has zero crossings at the right time to avoid crosstalk.

E. Bandwidth on Demand using Tree-Structured Filter bank

One of the attractive features of tree-structured filter banks is that the input may be at an intermediate point. For the tree with 3 levels used in this study, there are 8 possible inputs at the first level, 4 at the second level, 2 at the third and one at the fourth, the output. Every configuration is different, and each option may be represented with a schematic as shown. Here there are two low rate inputs to points 1 and 2, one double rate input to filter H2, one quadruple rate input to filter H3.

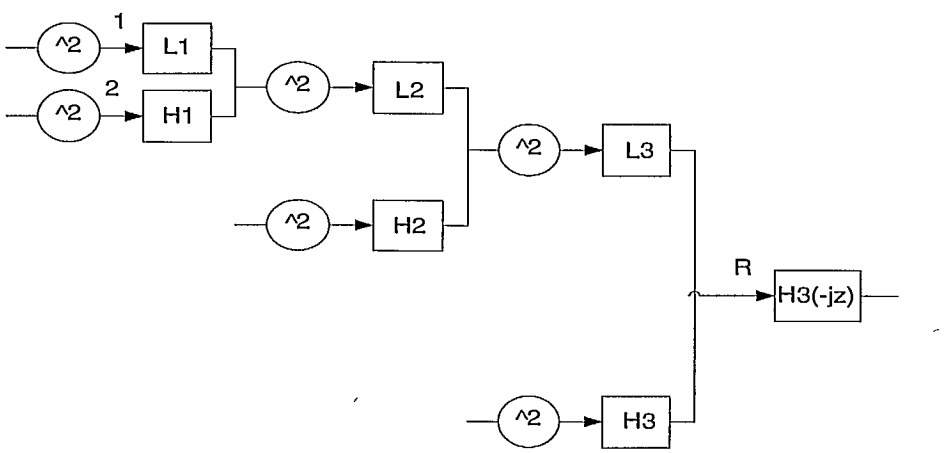


Figure 12 Possible configuration for entries into filterbank tree.

This configuration may be represented with a schematic as follows.

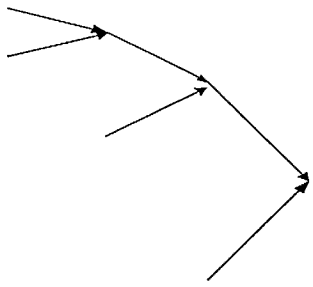


Figure 13 Simplified Schematic of the previous option. An interpolator/filter is represented with a directed line of the graph. This is a connected subgraph of the complete filterbank tree graph.

The number T_k of possible configurations at level k is given by the formula[9]

$$T_{k+1} = T_k^2 + 1 \qquad (3)$$

with T_0=1, T_1=2, T_2=5, T_3=26, etc. This formula results since the number of configurations at level k+1 is the straight-through configuration or one which has an arbitrary configuration from each of the subtrees of level k.

In system simulation it is difficult to keep so many possibilities under control, and therefore we have used another, equivalent approach. If data is to be inserted at a higher level, then we consider that it has been interpolated at a lower level with a time-division-multiplex filter. Thus, at each level, a filter pair may be either a *bone fide* filter pair, or a virtual filter that is a simple TDM multiplexer. In the schematic representation we use a dashed line for a virtual filter pair, and a solid line for an actual filter.

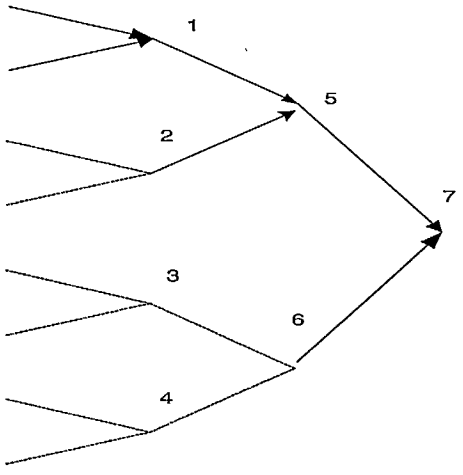


Figure 14 Simplified Schematic of the previous option. An interpolator/filter is represented with a solid directed line of the graph. An interpolator/multiplexer is represented with a dotted line.

Each configuration is labeled with a number in binary representation, the lower bit of which is 1 if a real filter pair is connected at node 1 above, and a 0 if a virtual filter is so connected, and so on for all 7 possible connection points. Certain of these 2^7 options are not valid if we wish to maintain an FDM arrangement, but there are 26 such arrangements as given in the next table.

The previous configuration is 1000101 low bit first , or 1010001 low bit last, or number 81 decimal.

In the implementation of the program, the configuration number is indexed with an integer from 1 to 26, and then the appropriate binary number is selected from the chart above. If a filter corresponds to a binary 1 then the usual filtering operations as described for 2-channel filter pairs is performed. If a zero, the input signal streams are multiplexed with a delay that is near the delay of the real filters, one delay being half a sample more, and one delay being half a sample less.

For example, filter f1 has length 12, order 11, and delay 11/2 or 5-1/2 samples. The interpolation using TDM then introduces one signal with a delay of 5 samples, and the other with a delay of 6 samples, and these outputs are, with zero padding at the end, entirely equivalent to the outputs from the real filters in terms of length.

Whether early or late depends on whether the filter in the real filtering case was low pass or high pass, and the same equivalence is used.

The advantage of this approach to handling the various configurations is that the signal processing with the input and output data samples is transparent to the configurations.

Table 3 Configurations for a 3-level tree-structured filter bank. For example valid(5)=82.

Configuration	Configuration number		Number of inputs at rate indicated.			
	Decimal	Binary: low bit first	Rate=1	1/2	1/4	1/8
1	0	0000000	1	0	0	0
2	64	0000001	0	2	0	0
3	80	0000101	0	1	2	0
4	81	1000101	0	1	1	2
5	82	0100101	0	1	1	2
6	83	1100101	0	1	0	4
7	96	0000011	0	1	2	0
8	100	0010011	0	1	1	2
9	104	0001011	0	1	1	2
10	108	0011011	0	1	0	4
11	112	0000111	0	0	4	0
12	113	1000111	0	0	3	2
13	114	0100111	0	0	3	2
14	115	1100111	0	0	2	4
15	116	0010111	0	0	3	2
16	117	1010111	0	0	2	4
17	118	0110111	0	0	2	4
18	119	1110111	0	0	1	6
19	120	0001111	0	0	3	2
20	121	1001111	0	0	2	4
21	122	0101111	0	0	2	4
22	123	1101111	0	0	1	6
23	124	0011111	0	0	2	4
24	125	1011111	0	0	1	6
25	126	0111111	0	0	1	6
26	127	1111111	0	0	0	8

F. Phase Synchronization in Rician channels

There are modulation choices that do not require phase synchronization, such as differential PSK, but they entail a performance penalty that is usually not acceptable for satellite mobile communications. Therefore it is necessary to establish a phase reference. With analog transmission of single sideband signals this was usually accomplished with pilot tones. However, digital modulation and signal processing allows other options.

Previous studies on pilot symbol-assisted synchronization.

Moher and Lodge [2] used pilot symbol-assisted modulation in connection with a study of trellis coded modulation for Rician fading channels. They point out that it is preferable to use embedded symbols rather than pilot tones, since embedded symbols cover the entire signal spectrum. Their paper concentrates mostly on the trellis coding. Their synchronization occurs as follows. A known pilot symbol sequence is embedded, with one pilot symbol for every M data symbols. At the receiver the received samples are modulated with the known reference symbols to produce a [noisy] channel estimate. The channel estimate is filtered with a filter that is approximately matched to the worst fading rate; the filter design is not specified further, but, in the absence of definition, it was likely a digital approximation of a raised-cosine Nyquist filter.

Cavers [3]has analyzed pilot symbol-assisted modulation for Rayleigh fading channels [2]. He makes the Nyquist interpolation filter satisfy Wiener filter design methods. He shows that the error rate has a break point as the pilot symbol spacing M is increased. The break point appears to be

$$Mf_D T = 0.5 \qquad (4)$$

Where f_D is the Doppler shift, T is the symbol period, and M the number of data symbols between each pilot symbols.

S Sampei and T Sunaga [4] investigated an arrangement in which the filter was a Lagrange interpolator of order 2 or 3. [For some reason they use the term Gaussian filter, not Lagrangian.] This appears to be a good choice if knowledge of the fading spectrum is absent.

Kim[5] compared the impact of using three filter types, a Wiener filter, like Cavers, a sinc filter (ideal brick wall) and a Gaussian (i.e. Lagrangian) interpolator. Kim showed that the pilot symbol spacing could be increased to the limit given by the previous equation for the Wiener and sinc filter (15), but the Lagrange interpolator was beginning to detract from performance at a pilot symbol spacing of 5 for the same parameters ($f_D T$=0.03).

Toshiaki Kuroda and Tadashi Matsumoto [6] have one of the few papers to address multicarrier synchronization. They consider a more general model, in which pilot symbols are inserted in each channel, at the same rate, but not at the same time. They use a Kalman filter for the channel model, in which they assume that the channel has the state space representation. Also, they use a two-parameter model for the channel, and derive from it a model for the subcarrier response. They stagger the pilots in each channel in order to give more frequency readings of the channel response. They compare their system with a per channel phase recovery of the same type, but in which the channel models are unrelated. Kuroda and Matsumoto discovered that sharing the pilot symbols improved performance over the per-channel synchronization procedure. However, they did not consider a scenario such as the one that we are using in which phase synchronization is performed only on a high speed channel.

G. Performance measures

The performance of the system with a variety of signal to noise ratios in a Rician channel is the primary output of the work reported here. The primary interest is the bit error probability.

There is also interest in the performance of the synchronization system itself, and thus the phase deviations between the channel and the channel estimate are measured and reported.

In such a synchronization system which functions by adding in a correction signal, there will be an extra power component. In a sense, this is also a measure of the correct choice of input bits to make the window signal vanish so that the pilot sequence may be injected into the window. If the choice of inputs were poor, the matrix required to solve a set of equations would be ill-conditioned, and a large correction signal would be needed.

Finally, it is of interest to determine the timing jitter inherent in this synchronization system.

H. Previous work on Tree-based filterbanks for communication

The concept of one broadband signal supporting a variety of bandwidths is not new. If we think of the electromagnetic radiation at one location we have such a signal.

When digital signal processing was introduced to communication one of the first large-scale applications was the transmultiplexer, which converted the single sideband signals for FM radio to digital signals for transmission in cities; these transmultiplexers involved uniform signals, and led to the discovery of the uniformly spaced (in frequency) channels of the polyphase filter bank. These concepts were also used in uniformly spaced multicarrier modems using offset-QPSK or vestigial sideband channels[14,15]. Similarly, OFDM communications using the fast Fourier Transform inherently involves uniformly spaced carriers.

The concept of different bandwidths was used in the digital analysis of signals destined for the human ear[7], since the filters in the ear have larger bandwidths at larger frequencies. This signal processing was often accomplished with 2-channel filter banks, one bank to analyze the signals, and another to synthesize a broadband signal based on the subchannel signals. It was been recognized inherently and explicitly[8] that any system that consists of a good [little alias, image or ripple] analysis/synthesis pair may also be used as a good [little crosstalk, ISI] synthesis/analysis pair for FDM communication.

The concept of wavelets was introduced to find a universal function of a variety of time scalings, usually dyadic, to analyze and synthesize signals. Soon, it was discovered, by using the more general trees discussed in section D, that a mix of bandwidths could be achieved; these are called wavelet packets in this literature. There are many references, and Zarowski [9] is recommended.

The concept of bandwidth on demand has a long history. Systems such as ALOHA packet communication and Ethernet are examples. These systems have relatively low efficiency. Time division digital transmission was attractive to support such services for wired applications; however, each user must recover the entire bit stream. The point must be stressed that the ability to change the traffic mix easily, and electronically, is required.

Although finite duration wavelets or wavelet packets cannot be used for communication because the frequency characteristics are unsatisfactory, a number of researchers in the mid 90s began to explore applications of some concepts developed in wavelet research, involving what they called 'arbitrary tiling of the time-frequency plane' but which we may think of as FDM communication supporting a variety of data rates in a spectrally efficient means. This paper is an outgrowth of this work.

The concept appears implicitly in work by Sablatash [10] and Lindsay [11,12]. The first step in our work[13] involved showing that the original concepts, based on 'wavelet packets', which produced two

non-adjacent sidebands , required modification with the introduction of a vestigial single sideband filter, to produce vestigial sideband, or equivalent [16] offset QPSK signals.

The next step [17] in our work involved showing that the concept of using the *same* universal filter at every level of a tree-based filter bank, as in wavelet theory, was inappropriate for this application; a different filter should be used at each level, with those at the next level having half the transition bandwidth.

Then in [18,19,20] we showed that the use of polyphase filterbanks instead of VSB filters gave system efficiencies. Recently [21], the sensitivities of such systems under a variety of timing and phase errors were determined. Also, the sensitivity of the phase recovery process to offset QPSK or vestigial sideband was studied [21].

This paper continues in this line of activity by showing that linear phase filters are very useful in this application, and the paper also analyzes the performance of a synchronization arrangement proposed by John Lodge[1].

II. Description of Simulation

A. Introduction

The system is evaluated by computer simulation. Here is a simple explanation. Data symbols are chosen randomly and passed through a tree-structured filterbank. The location of the symbols where the pilot sequence is to be inserted in made clean, and the pilot symbols are inserted. The resulting real signal is passed through an offset QPSK modulator, and a lowpass root-Nyquist filter. The output models the channel input, and this is modulated to simulate a Rician channel, and noise is added. At the receiver, this signal is passed through a lowpass root-Nyquist filter, and then phase is estimated. The phase estimates are used to reduce the phase shift caused by the Rician channel. Then the location of the pilot sequence is made clean, and the real part of the phase-adjusted signal is made and passed to the receive filterbank. The program compares the transmitted data with the recovered data, and evaluates the error rate.

The simulation is written in the programming language MATLAB, and the program listing appears at the end of this report.

The basic structure of the program is as follows

```
initialize
for every Doppler shift
    for every SNR
      calculate error rate
      calculate phase error
      calculate potential timing error
    end
  plot a line of the error rate/SNR curve
end
cleanup
```

The phase and timing variances are available for plotting or analysis.

The program is described as follows in this section. A general description has been provided in this introduction. The details of the processing in the filterbank are provided in section II B. The details of the phase estimation is provided in section II C. The Rice modeling is a straightforward application of Richard Young's existing CRC MATLAB routines, and is described in Appendix 2. The commented source files are appended.

B. Tree-structured filter bank and pilot sequence insertion

The initialization portion of the program creates six sets of filter coefficients that are used in the simulation to represent the virtual filters, labeled t1, ht1, t2, ht2 and t3, ht3. These are created in the routine `maketree`, which also reads the filter coefficients from a file, calls them f1, f2 and f3 and creates hf1, hf2 and hf3 as the high pass versions.

As indicated in section I, in order to avoid writing 26 simulation programs, one for each configuration, the simulation is based on the concept of presenting a block of data to the input of the tree, for example a block 20 by 8. Each of the 8 vectors of length 20 is presented to a filter input, and the choice as to whether to filter the data or pass it through unaltered is made to depend on a 7-bit number called configuration. This 7-bit number is used to define 7 Boolean variables {is1a, is1b, is1c, is1d, is2a, is2b, is3a}, where is1a is true if a real filter is to be used and is false if a virtual filter is to be used for the first of the four filter pairs

appearing at the input. The variables is1x are used for the first level of the tree, the inputs, is2a and is2b for the second level and is3a for the third level.

The program `lintree` which performs the filter tree calculation is then just a sequence of if statements depending on the variables isyx. For example, the first set of statements is

```
if is1a
"
   sa=upfirdn(databl(:,1),f1,2,1)+upfirdn(databl(:,2),hf1,2,1);
"
else
"
   sa=upfirdn(databl(:,1),t1,2,1)+upfirdn(databl(:,2),ht1,2,1);
end
```

Here databl(:,1) is the first vector of input data, and databl(:,2) is the second. The output from the 2-channel filter pair composed of {f1,hf1} or {t1,ht1}, depending on `is1a`, is calculated using the MATLAB function `upfirdn` which performs an upsampling of the input data by 2.

The rest of the program `lintree` simply completes the calculation.

The program `detree` performs the inverse function to `lintree`; given an input signal `sigtotal`, it derives the set of 8 output vectors. There are only two complications here. In the one, we must ensure that the data samples are taken at odd sample times to avoid crosstalk. But the function `upfirdn( , ,1,2)` which performs a decimation, does not allow the control of the phase of the decimation. This must be done by padding the input with a zero sample, which is the first line of `untree`.

The only other matter of concern is to make the correct choice of sign of the real quantity hg that appears in the description of 2-channel filterbanks. This is done with the correct choice of signs as exhibited in the program.

A typical `if` statement in `untree` is the first

```
if is3a
   data1=[upfirdn(sigtotal,f3,1,2) -upfirdn(sigtotal,hf3,1,2)];
else
   data1=[upfirdn(sigtotal,ht3,1,2) upfirdn(sigtotal,t3,1,2)];
end
```

and here the input `sigtotal` is used to prepare a matrix with two elements, using the MATLAB function `upfirdn`, and choosing the appropriate filter pair depending on the truth or falsity of `is3a`. The rest of the routine proceeds in working its way away from the root of the tree.

The program `makeAmatrix` uses `lintree`. We define the A matrix as the response in the window of an input at one of the 4 by 8=32 positions for the input positions where the correction samples are going to be introduced. But all we really need is the response to an input at each of the 8 input channels, since the other elements of the A matrix may be obtained by shifting. Thus the routine `makeAmatrix` simply builds up the A matrix for every channel by computing the response to a 1 in that channel and a zero elsewhere, and examining the response at the output of the tree using the routine `lintree`. The details of filter type are hidden from the routine, making it configuration independent.

The program that performs the creation of data and the insertion of the augmented m-sequence is `addseq`, which we now describe. The first statement creates a data block of length 16 by 8, and saves a copy for use

in the receiver to calculate the error rate. Then 4 zeros are inserted in the middle of each block of data, one for each channel. This augmented datablock is passed through the tree, and the quantity which we described as spillover in the introduction and in the program, is found by taking 32 samples from the output of the tree at the position of the window. We then find the correction samples, that are to appear at the input, by the equation

```
correction=A/spillover.
```

These desired samples are formed into a data block suitable for the filterbank, and passed through the filterbank using the routine `lintree`. This output, suitably delayed, is subtracted from the previous output, thereby creating the blank. Finally, the augmented m-sequence, which is the vector `mseq` in the program is inserted. Here are some illustrations of typical sequences.

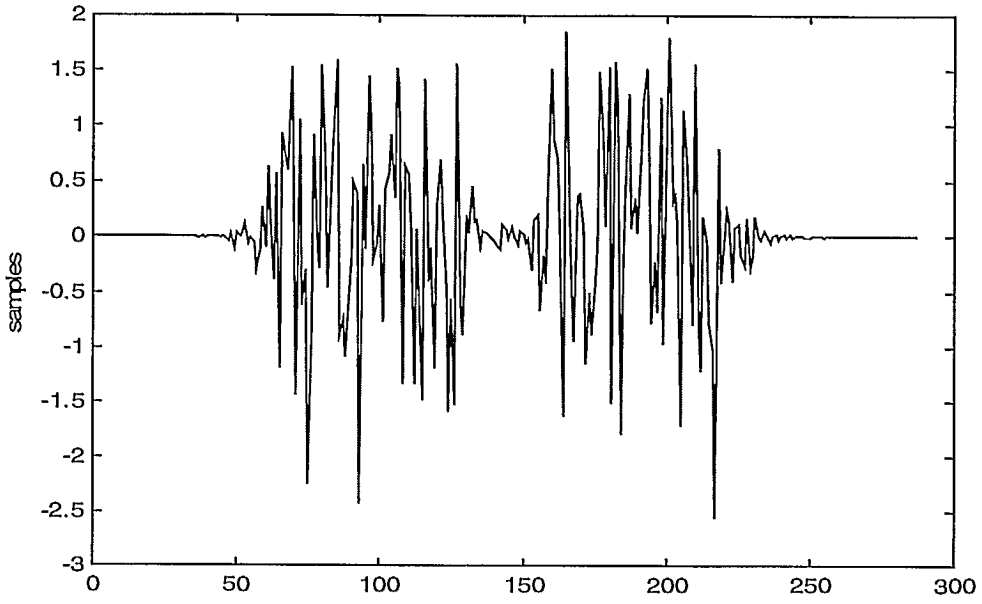


Figure 15 Output of filterbank when the middle 32 inputs have been made zero. Not that in the window the output is small, but not zero.

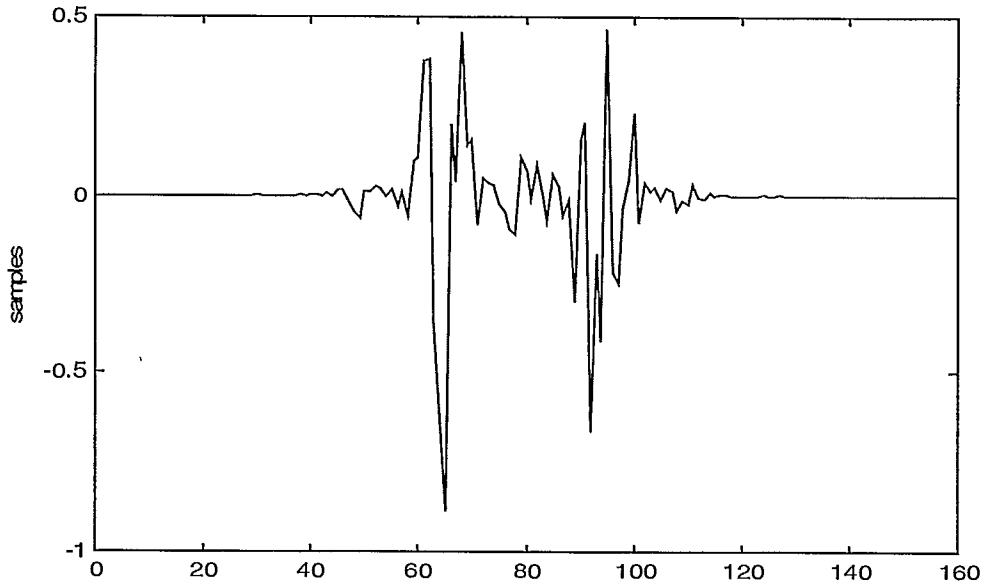


Figure 16 Output from filterbank when the correction signal has been applied to the input. This signal is to be subtracted from the previous figure, when suitably delayed.

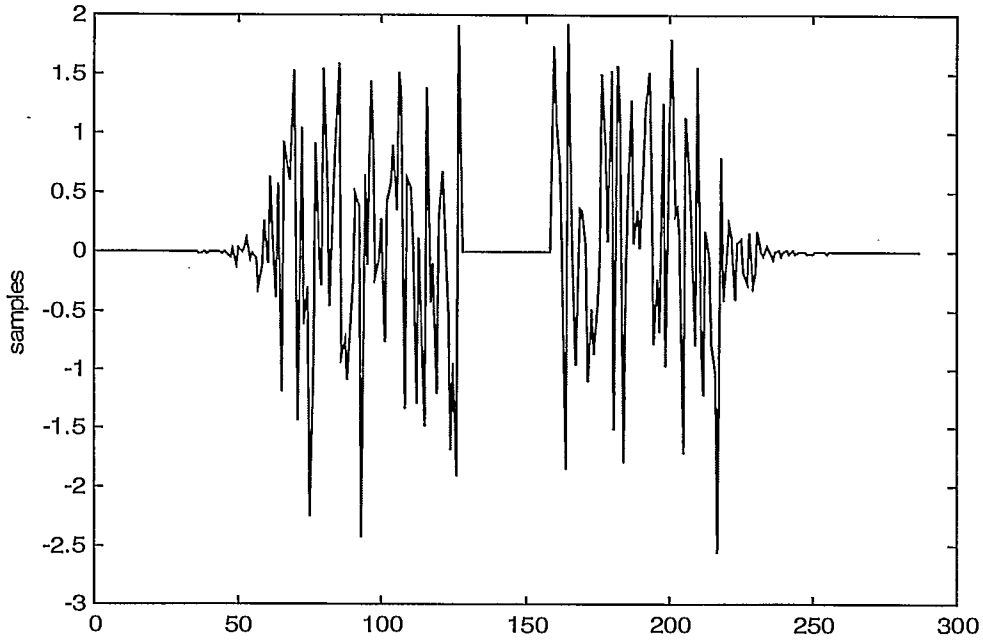


Figure 17 After subtraction, the position where the pilot sequence is to be introduced has been cleaned out.

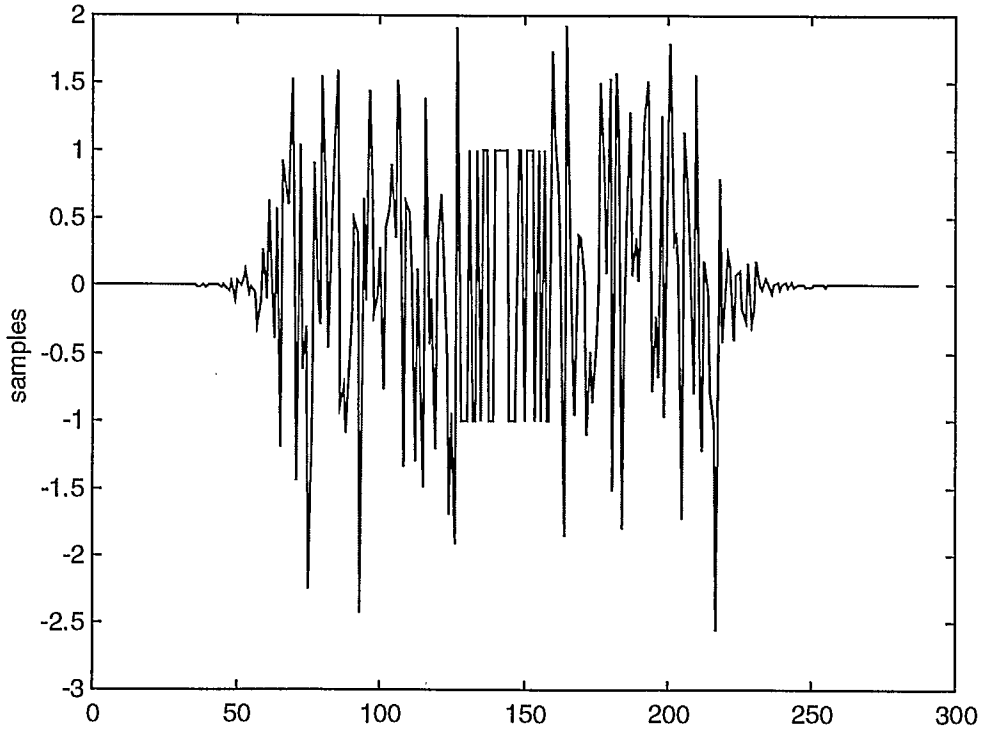


Figure 18 After the pilot sequence is added in, this is the output from the filterbank

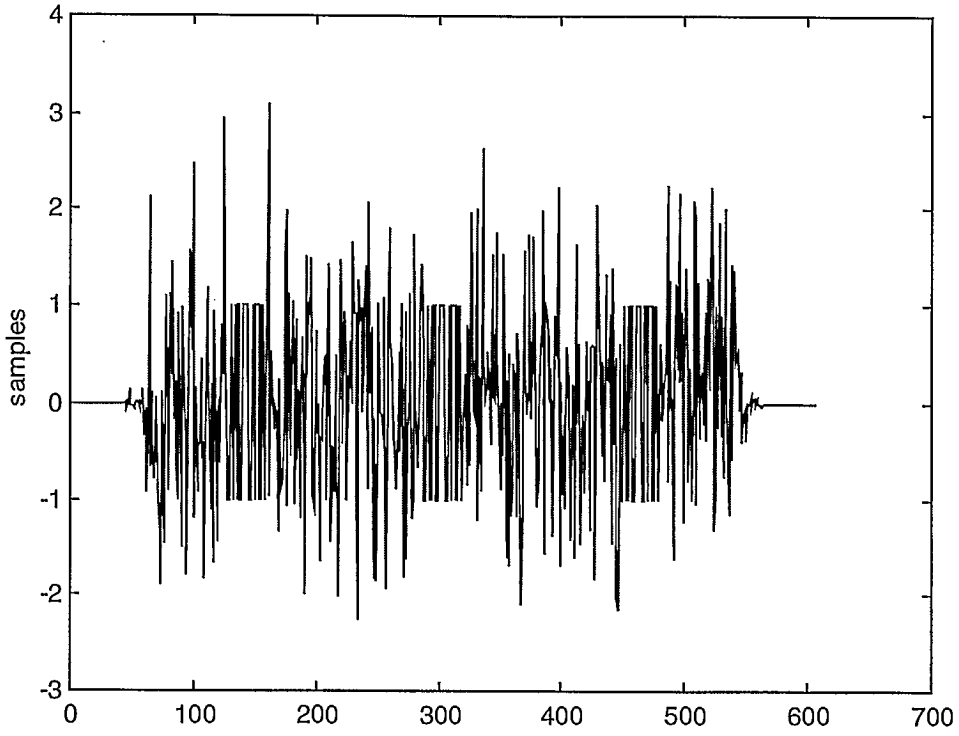


Figure 19 If the simulation involves more blocks, they are calculated, delayed, and added.

It is a measure of the utility of this concept to determine the eigenvalues of the matrix AAT used to determine the eigenvalues of the matrix AAT. The ratio of the maximum to minimum is presented in Figure 20.

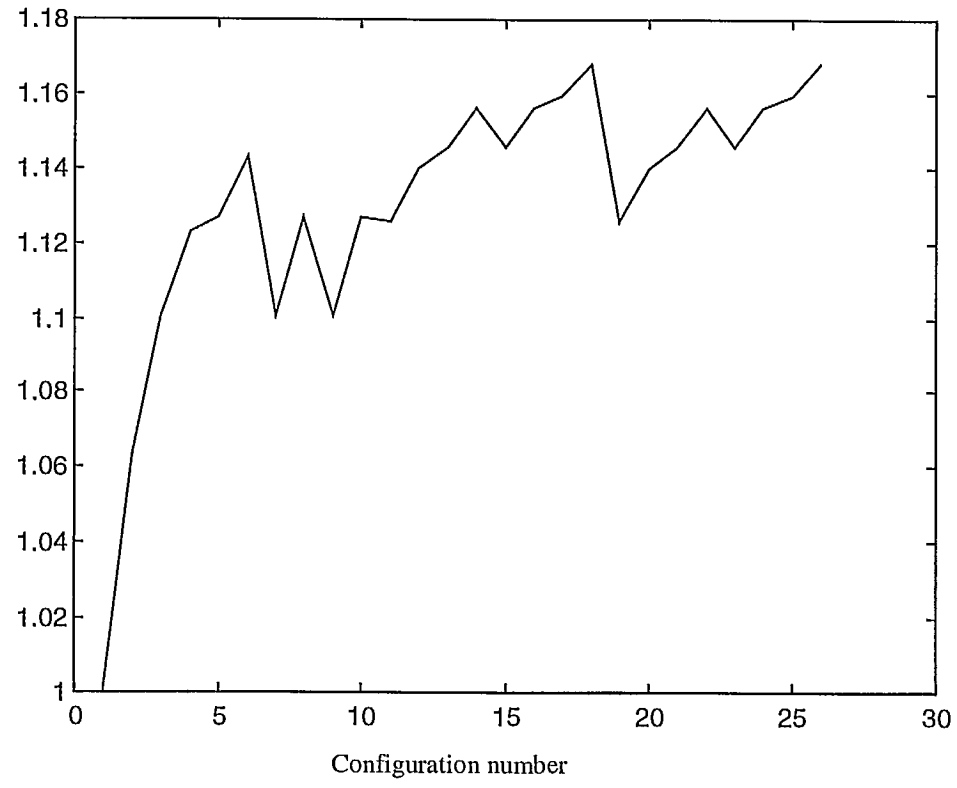


Figure 20 Plot of ratio of maximum to minimum of the square root of the eigenvalues of ATA for the 26 bandwidth-on-demand configurations.

In this section an outline has been presented on the filtering portions of the simulation program.

C. Synchronization system

Synchronization is based on the use of pilot symbols. A real data sequence containing a periodic pilot sequence is passed through a staggered QAM modulator. The resultant signal is passed through a lowpass root-Nyquist half-band filter. The signal is then modulated with a random sequence that represents the action of a Rician channel. Next, complex Gaussian noise of the requisite power is added to form the received signal. The actions of the receiver are the following. The received signal is passed through a lowpass root-Nyquist half-band filter used at the transmitter. Then the signal is demodulated with an offset QPSK demodulator. The resulting signal is passed through the matched filter of the pilot sequence. The output from the pilot sequence is determined at the center of the matched filter output, periodically, and forms an estimate of the Rician channel and the offset QPSK modulator/demodulator. This channel estimate is formed conceptually by passing the samples through an interpolating filter which should be an M-band full-Nyquist filter but which we have approximated with a Lagrange interpolation filter. Then the conjugate of the channel estimate multiplies the data sequence, and the real part of the resulting sequence is taken, and the locations where the pilot symbols were inserted are removed. This real signal, with the position of the pilot symbols removed, is fed through the filter tree receiver, and the received data is recovered.

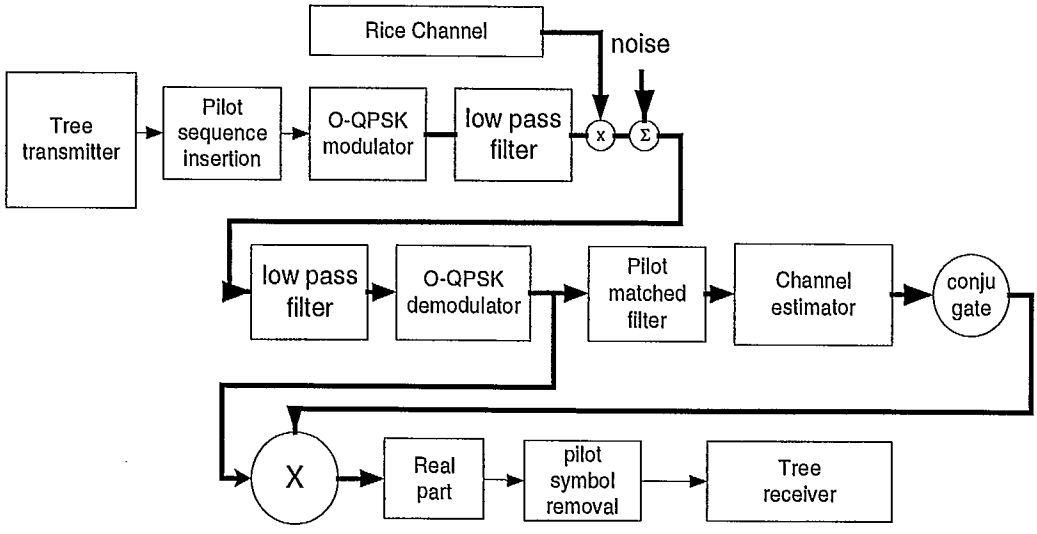


Figure 21 Synchronization system.

In the simulation, we model the system with three blocks of data, each of which consists of 128 positions for data and 32 positions for the pilot sequence, or a total of 3x160=480 samples in all. The operation of offset QPSK modulation does not change the length of the data sequence, but the filters do increase the length.

In the receiver, we throw away the first and third outputs, and only examine the error rate for the middle block of data.

In the program, the synchronization is first performed in the main program rsim3 by the use of the pilot sequence and random data filtering routine `addseq`. This is called three times, and a composite signal illustrated in the previous section is made. This 3-block signal is passed through the channel with a call to the routine `makesqam` which does an offset QPSK modulation, convolved with the low pass filter f3, modulated with a Rician channel generated in routine `rice3`, and then has Gaussian noise of the correct magnitude generated and added in routine chancgn. The received signal is filtered with the root-Nyquist filter f3 and converted to baseband using the offset QPSK demodulator `makesqam` again.

The phase estimation is first performed in the routine `getphase4`. Here the received signal is simply filtered with the matched filter to the augmented m-sequence forming the pilot sequence, and sampled at the correct times; this produces three output, `chanest1`, `chanest2` and `chanest3`, one for each of the three blocks. As well, an estimate of the peak time is made and returned as `t1`, `t2` and `t3`, but these are not currently used.

The other routine used to complete channel estimation is `myinterp4`. This performs a Lagrange interpolation using the three channel samples obtained in getphase4, and yield an estimate for the channel response.

III. Performance

A. Error rate vs SNR, Doppler, number of sync words used (1 or 3)

The major use of the simulation is to determine bit error rate (BER). In this section we present some results of these simulations. For each plot there are two solid theoretical lines for reference. The first, the lower, is the error rate for BPSK in a flat channel, and the other line without a symbol in the plots is the reference for Rician channel, perfect phasing, based on the theory of the Appendix.

The SNR quoted is similar to Eb/N0 with the following differences:
-the signal power is relative to the power of the direct path in the Rician channel; the extra power available in the Rayleigh component has not been included.
-the effective data rate 128/160 has not been used to adjust the definition of Eb.
-the increase in signal power due to the synchronization.

All these factors make the plotted SNR larger than Eb/N0 by 1.81 dB for K = 7 dB, consisting of 10 $\log_{10}(160/128)$ = 0.97 dB, and 10 $\log_{10}$ (1+10^(-K/10))= 0.79 dB, and 0.05 dB for synchronization power.

The figures are summarized in the following table

Table 4 Figure numbers for various error rate calculations.

Doppler (Hz)	Interpolation	Figure: Configuration 127	Figure: Configuration 82
100, 200, 300, 400, 500	Yes	23a	24a
100, 200, 300, 400, 500	No	23b	24b
100, 300, 1000, 3000, 10000, 30000	Yes	23c	24c
100, 300, 1000, 3000, 10000, 30000	No	23d	24d

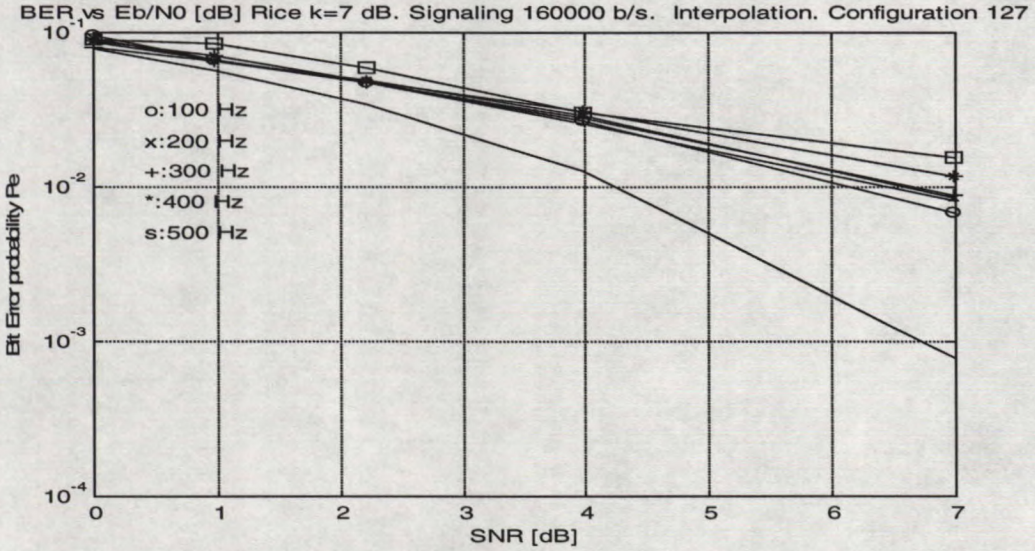


Figure 23a Bit Error Probability for Rician channel K=7, Doppler 100, 200, 300, 400 and 500 Hz with signaling rate 160 kb/s, for configuration 127.

From Figure 23a, the results for Doppler of 100, 200 and 300 Hz are about the same, showing that Doppler in this range has little effect on performance.

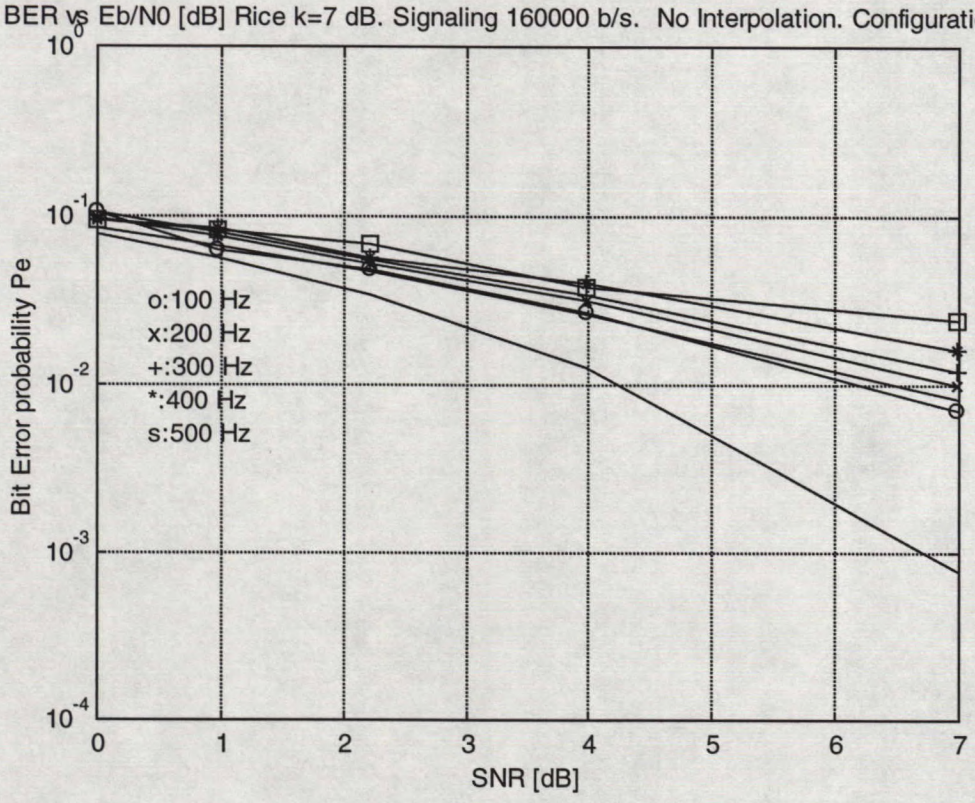


Figure 23b Bit Error Probability for Rician channel K=7, Doppler 100, 200, 300, 400 and 500 Hz with signaling rate 160 kb/s, for configuration 127, with no interpolation.

From Figure 23b, the performance is not as good as the case with interpolation.

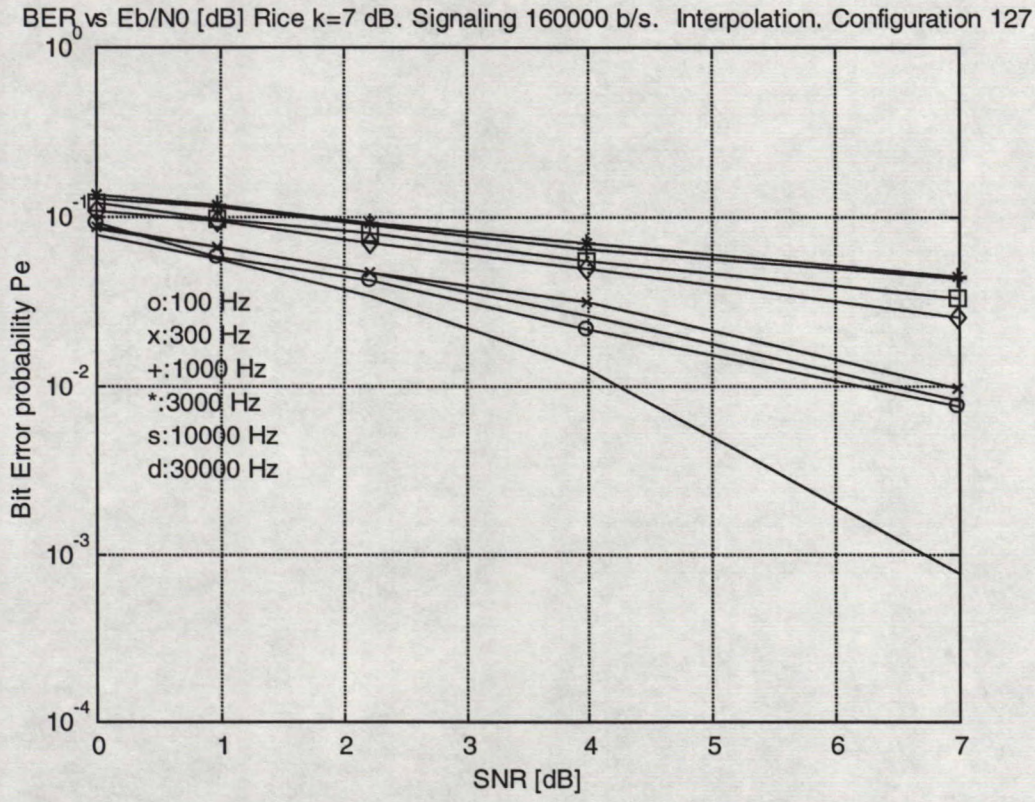


Figure 23c Bit Error Probability for Rician channel K=7, Doppler 100, 300, 1000, 3000, 10000 and 30000 Hz with signaling rate 160 kb/s, for configuration 127, with interpolation. The error rate is worst with a Doppler of about 3000 Hz, about the bandwidth the receive filter; for larger Doppler the error rate decreases slowly with Doppler. In the Doppler legend s means square and d diamond

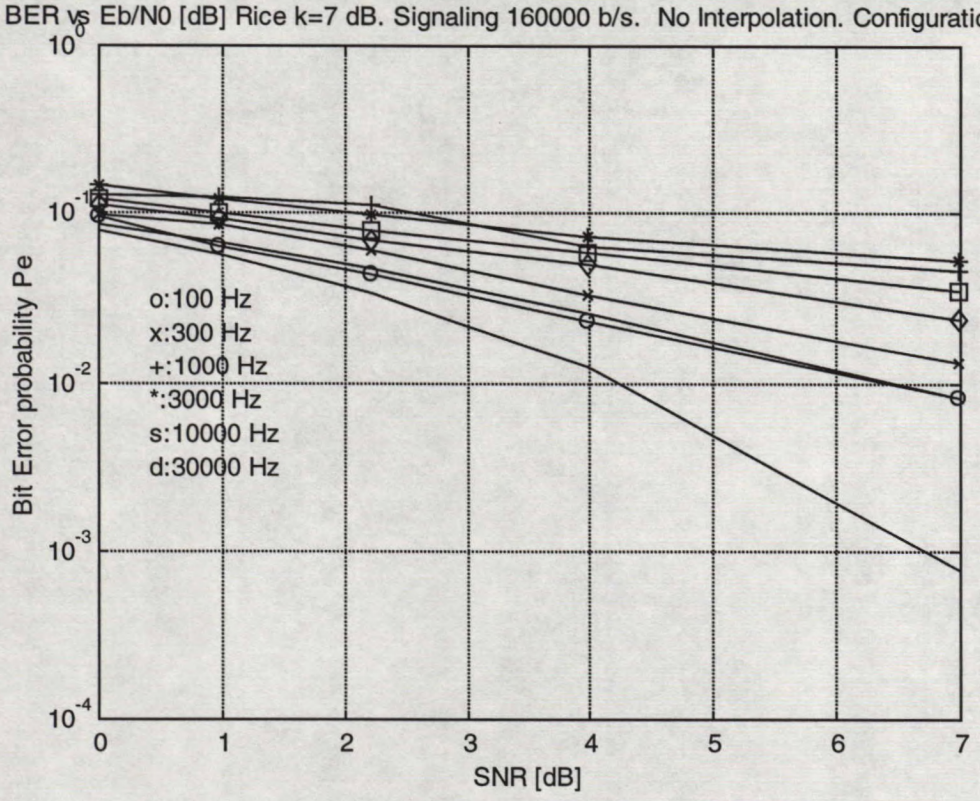


Figure 23d Bit Error Probability for Rician channel K=7, Doppler 100, 300, 1000, 3000, 10000 and 30000 Hz with signaling rate 160 kb/s, for configuration 127, with no interpolation. The error rate is worst with a Doppler of about 3000 Hz, which just fits in the receive filter; for larger Doppler the error rate decreases slowly with Doppler. In the Doppler legend s means square and d diamond.

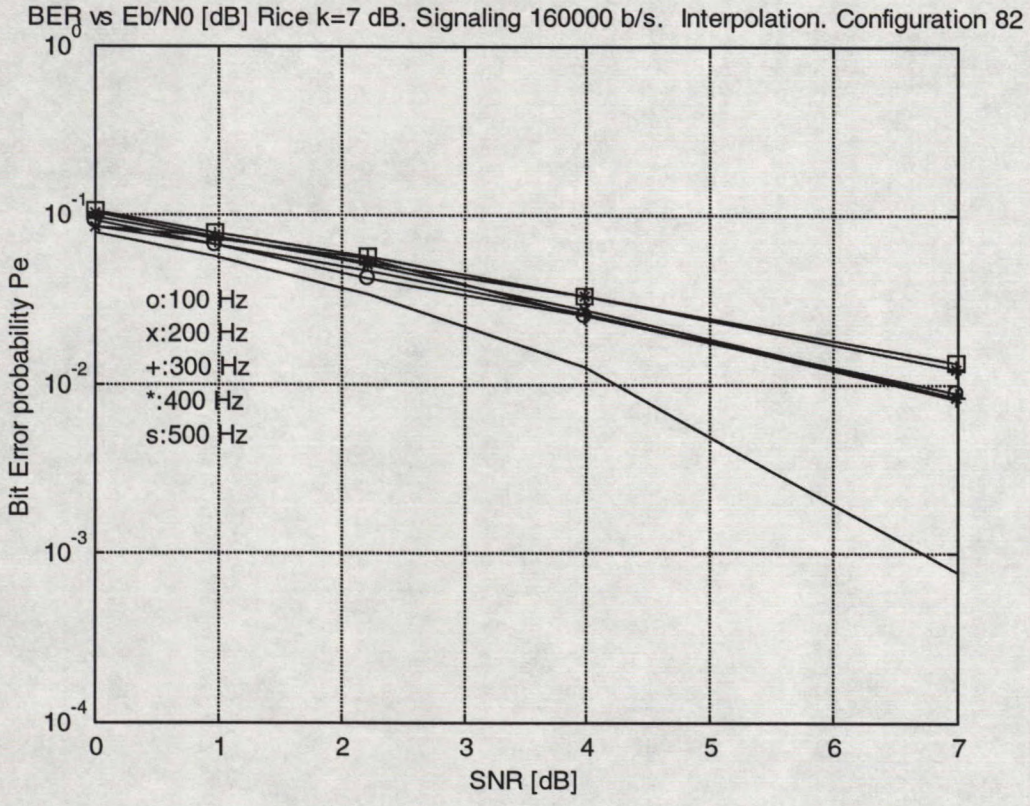


Figure 24a Bit Error Probability for Rician channel K=7, Doppler 100, 200, 300, 400 and 500 Hz with signaling rate 160 kb/s, for configuration 82.

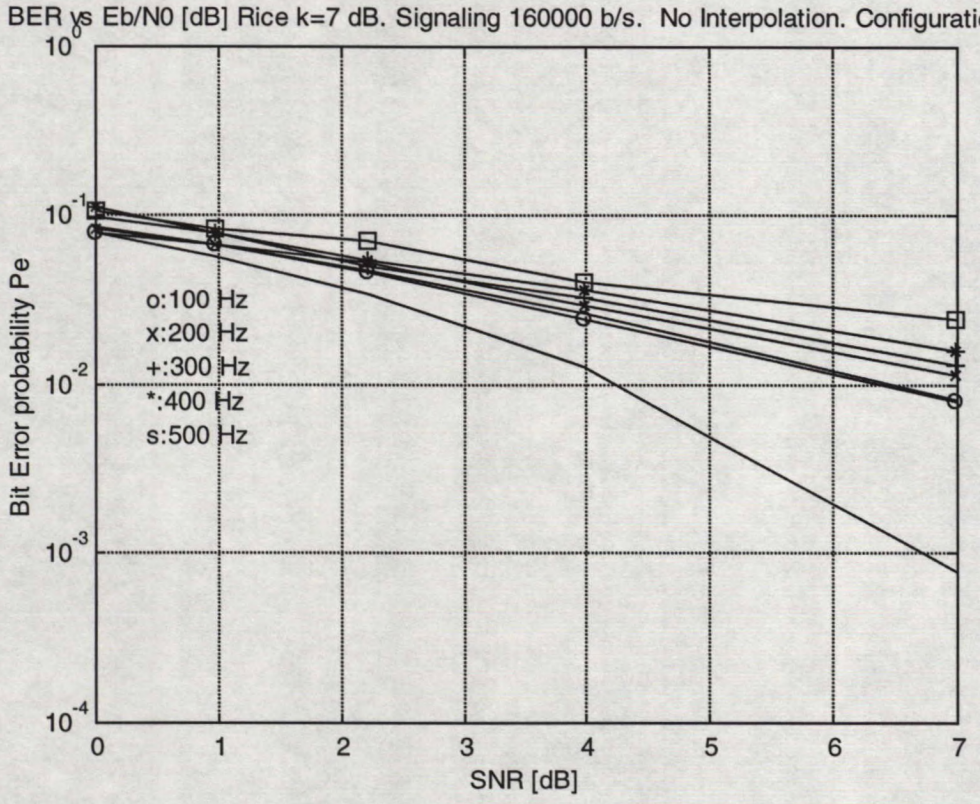


Figure 24b Bit Error Probability for Rician channel K=7, Doppler 100, 200, 300, 400 and 500 Hz with signaling rate 160 kb/s, for configuration 82, with no interpolation.

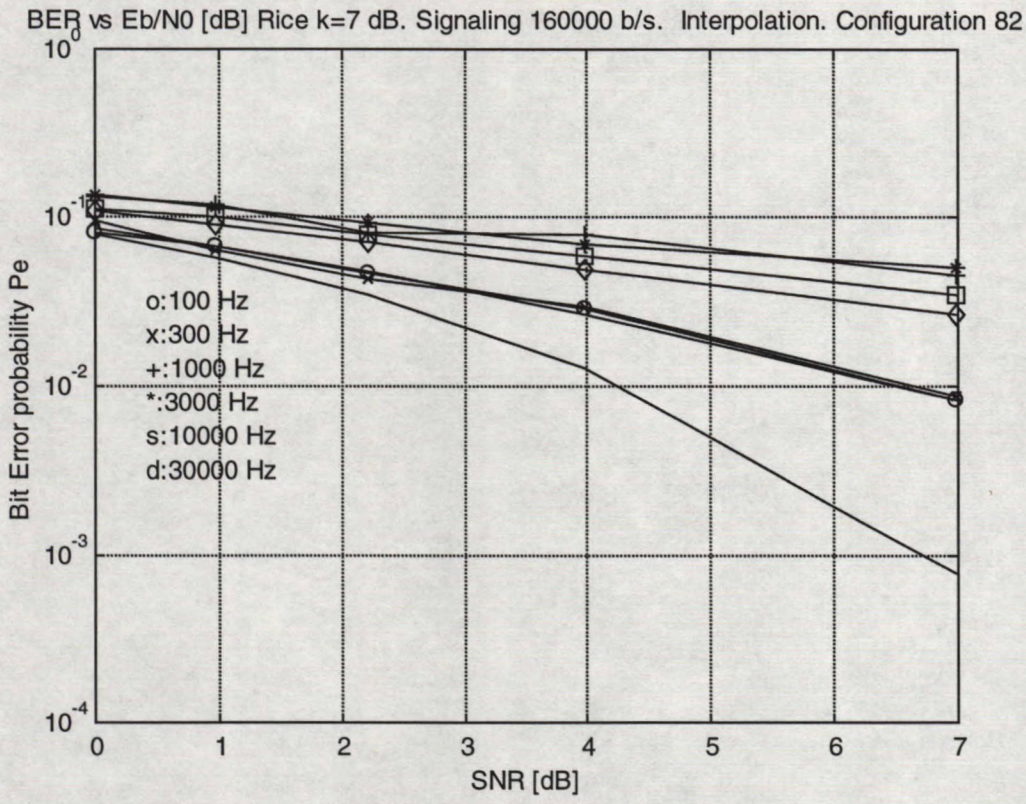


Figure 24c Bit Error Probability for Rician channel K=7, Doppler 100, 300, 1000, 3000, 10000 and 30000 Hz with signaling rate 160 kb/s, for configuration 82, with interpolation. The error rate is worst with a Doppler of about 3000 Hz, about the bandwidth of a receive filter; for larger Doppler the error rate decreases slowly with Doppler. In the Doppler legend s means square and d diamond

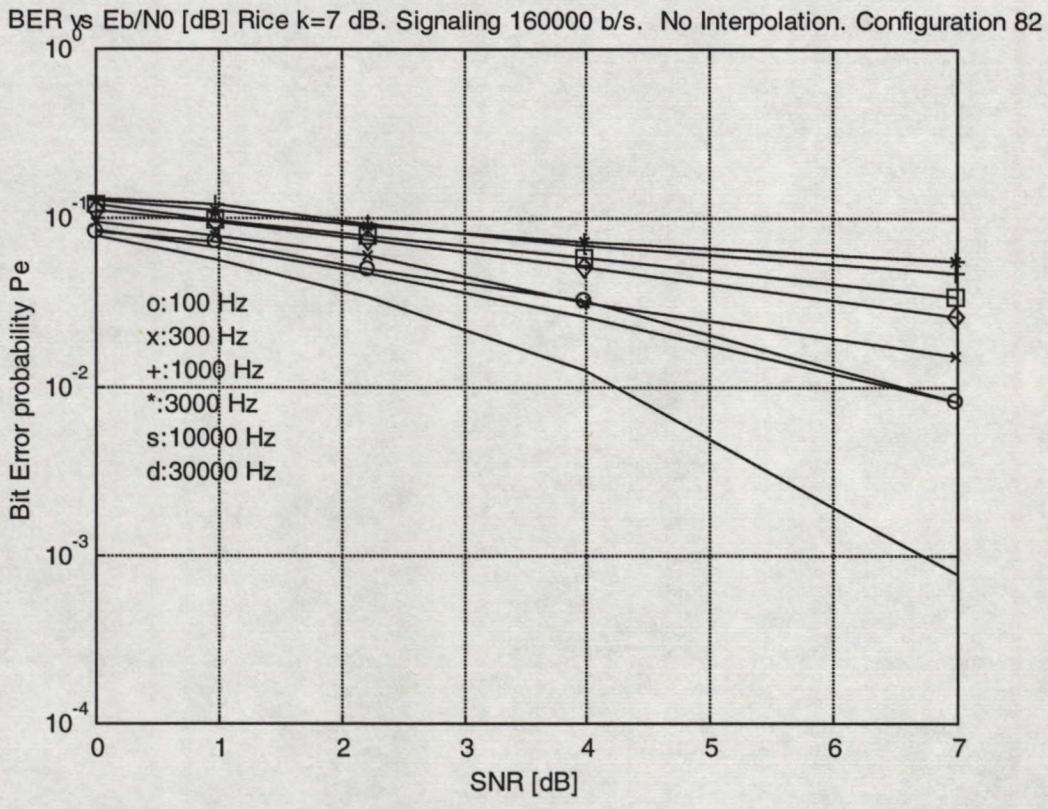


Figure 24d Bit Error Probability for Rician channel K=7, Doppler 100, 300, 1000, 3000, 10000 and 30000 Hz with signaling rate 160 kb/s, for configuration 127, with no interpolation. The error rate is worst with a Doppler of about 3000 Hz, which just fits in the receive filter; for larger Doppler the error rate decreases slowly with Doppler. In the Doppler legend s means square and d diamond

B. Synchronization power

It requires extra power to generate the correction signal. One measure of the power is to calculate the average power appear in the window. A theory has been determined for this function, outlined in Appendix 3, and we have compared the experiment to the theory in the accompanying figure.

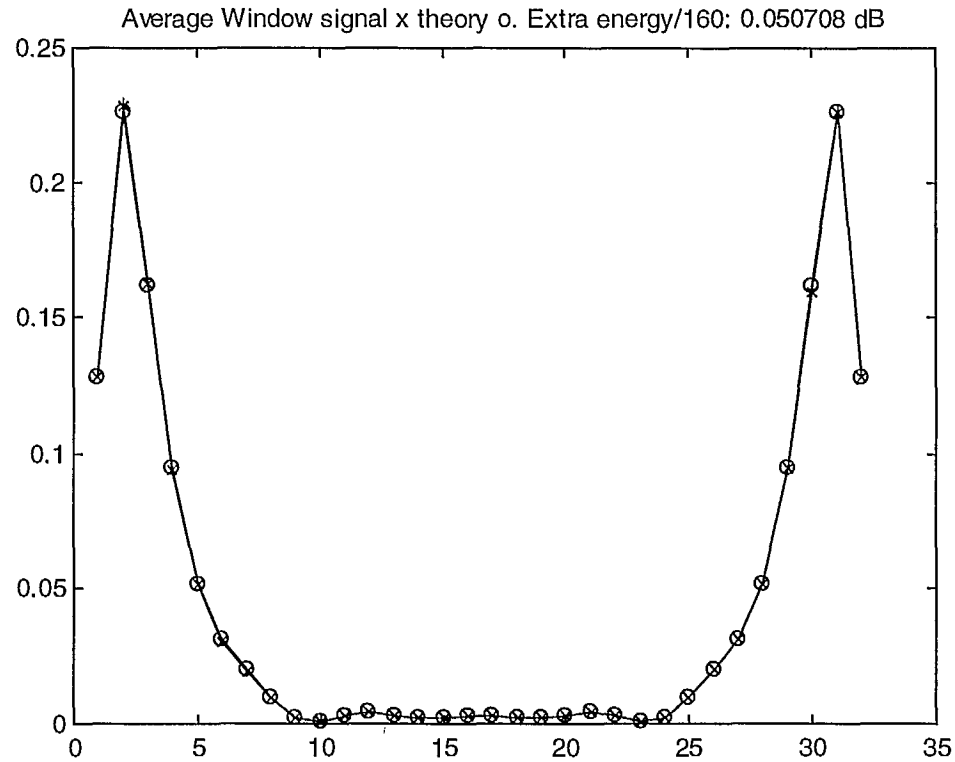


Figure 24e Plot of the signal energy in the window, as a function of the window sample, and the diagonal of AA^T-1, for configuration 117. The average is over 28666 runs.

The agreement is excellent.

C. Phase Jitter

In order to determine the performance of the phase synchronization system we have also monitored the difference in phase between the channel estimate and the channel, as a function of signal to noise ratio. The result is in the accompanying figure.

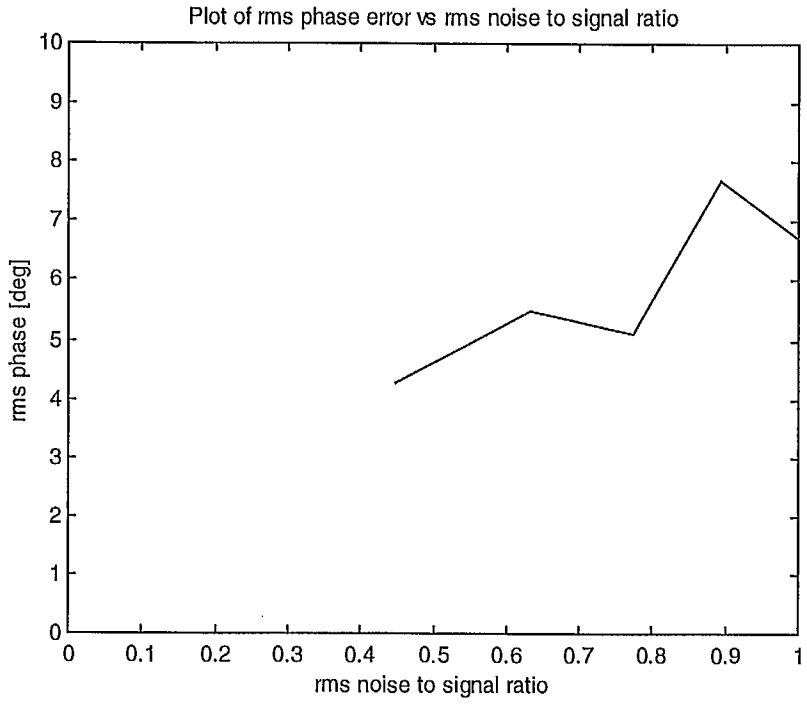


Figure 25a Simulated rms phase noise across the middle data block, with interpolation of the phase estimates from adjacent pilots. Doppler channel 100 Hz bandwidth, data rate 160 ks/s. Theory indicates that the phase noise should be proportional to the noise to signal ratio, as indicated.

Based on this measurement, for the Doppler indicated, the phase noise is what is to be expected.

The next figures give the phase noise and timing error for configuration 82 when there is no interpolation.

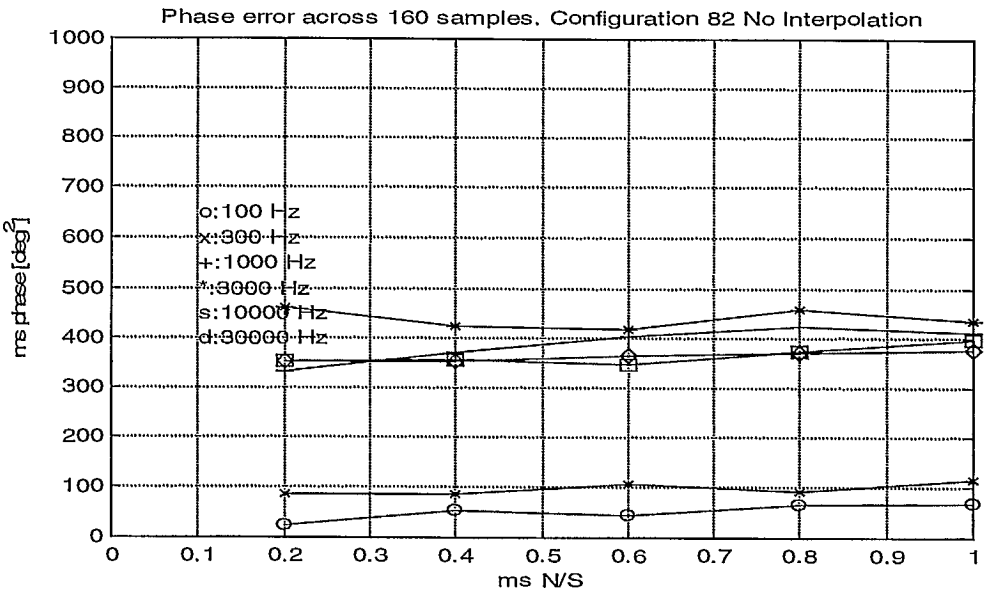


Figure 25b Mean squared phase noise for various Doppler frequency shifts, signaling frequency 160 k/s

Based on the results of Figure 25b, the phase noise appears to have a component that depends on the Doppler frequency shift, and a component that depends on the added noise level.

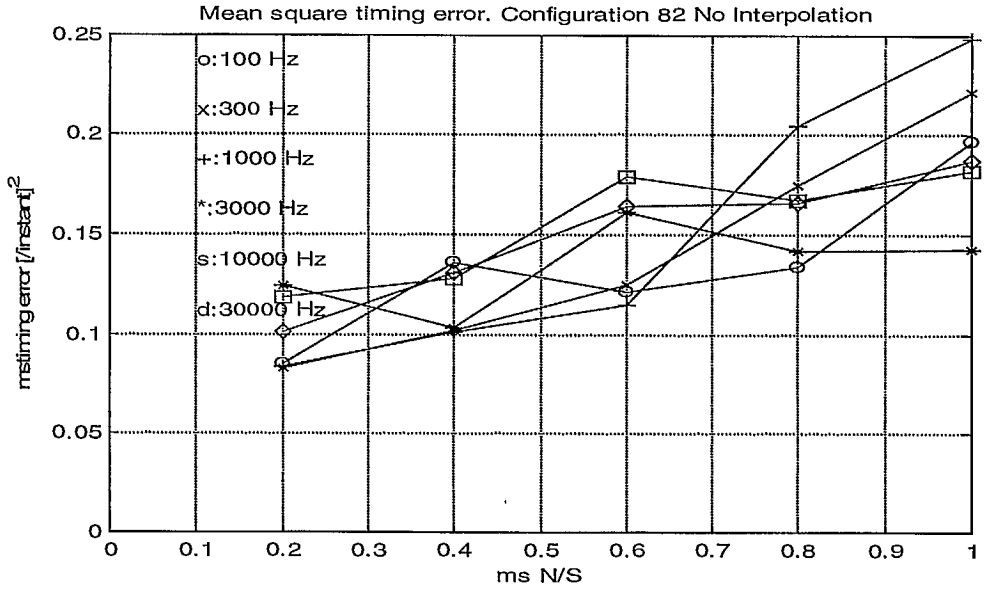


Figure 25c. Mean square timing error for various Doppler frequency shifts, signaling frequency 160 k/s.

The timing error does not appear to depend strongly on Doppler shift, but more on the additive noise. As well there appears to be a static offset.

IV Simplified Receiver

When the channel responses are observed, it appears that when the bandwidths are chosen according to the rules previously outlined, that the channel responses are much the same, except shifted in frequency. Therefore it has been conjectured that a simple receiver could be used, capable of receiving only a single channel at a time.

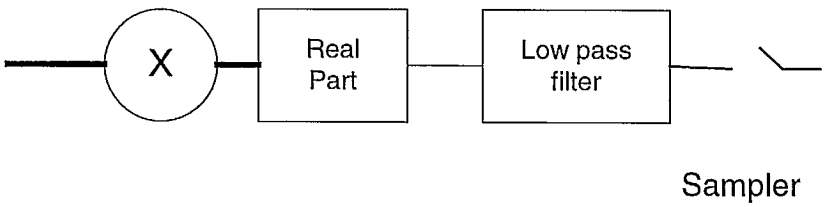


Figure 26 Simplified Receiver

Thus, the receiver, instead of the tree-structured filter bank, would have a single channel receiver. It would always be possible to make a simplified receiver in which the receive filter coefficients were modified from channel to channel, but the point is whether the same filter may be used. This has not been evaluated in a simulation, but we have calculated the response of such an arrangement, and conclude that such a simplified receiver is indeed possible when these linear phase filters are used.

The concept of a simple single channel receiver is tested by examining the pulse response from 16 channels into channel 1. This is done simply by modulating the response to that of the first channel, and using as a receiver the first channel receiver. The linear phase filters f1, f2 and f3 are used. We measure the intersymbol interference for each configuration, and this is plotted in the figure. As noted, intersymbol interference is negligible for radio systems with low SNR, being of the same order as the interchannel crosstalk power [40 dB]. The pulse response is normalized to unity.

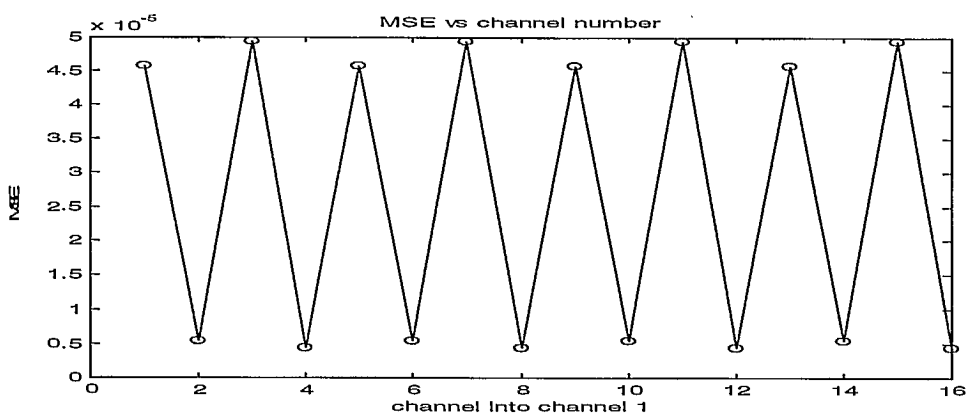


Figure 27 The residual intersymbol interference when transmitting from the indicated channel into a receiver tuned to channel 1. The mean squared error measures the residual, and these results indicate that the ISI is neglibible.

Based on these results, there is a simple single-channel receiver based on the linear-phase prototype filters.

V. Conclusions and Further Work

The simulation results suggest that Lagrange 3-point interpolation will provide performance for the range of Doppler shifts in the range 0 to 300 Hz that are essentially the same as the ideal Rician channel receiver. The simulation demonstrates that filter designs consistent with the parameters of the study in terms of number of channels and pilot rate are feasible. A simplified receiver appears feasible.

Work to be done includes the following

-analyze all configurations.
-speed up simulation by calculating Lagrange interpolation coefficients once instead of every iteration.
-analyze feasibility of replace Lagrange interpolator with Nyquist filter with better stopband performance.
-use stored random IIR initial conditions for the Rician channel modeling to avoid calculation of Rician variates that are junked.

The simulation does not consider acquisition.

Include error control coding to improve error rate performance.

VI: Acknowledgements

Thanks to John Lodge, Mike Sablatash and Paul Guinand for discussions on this subject, particularly in connection with modeling Rician channels, and Mike Sablatash for assistance in editing this report. The financial support under Industry Canada's Spectrum Research Funding is gratefully acknowledged.

VII References.

1. John Lodge, Work Statement, Feb 1998.

2. Michael L. Moher and John H. Lodge, "TCMP-A modulation and Coding Strategy for Rician Fading Channels", *IEEE Journal on Selected Areas in Communications*, Vol. 7, No. 9, Dec 1989, pp 1347-1355.

3. S. Sampei and T. Sunaga, "Rayleigh fading compensation method for 16QAM in digital land mobile radio channels", *IEEE Vehicular Technology Conference*, San Francisco, CA, May 1989, pp. 640-646.

4. J. K. Cavers, "An Analysis of pilot symbol-assisted modulation for Rayleigh fading channels, *IEEE Transactions on Vehicular Technology*, Vol. 40, Nov. 1991, pp 686-693.

5. Toshiaki Kuroda and Tadashi Matsumoto, "Multicarrier Signal Detection and Parameter Estimation in Frequency-Selective Rayleigh Fading Channels", *IEEE Transactions on Vehicular Technology*, Vol. 46, No. 4, Nov. 1997, pp. 882-890.

6. Young-Su Kim, "New Rayleigh Fading Channel Estimator Based on PSAM Channel Sounding Technique", *1997 IEEE International Conference on Communications,* pp 1518-1520

7. Ronald E. Crochiere and Lawrence R. Rabiner, "*Multirate Digital Signal Processing*", Englewood Cliffs: Prentice-Hall, 1983.

8. M. Vetterli, " Perfect transmultiplexers," *Proc. 1986 IEEE Intl. Conf. Acoustics, Speech, and Signal Process. (ICASSP '86), Tokyo, Japan*, 1986, pp. 2567-2570.

9. C. J. Zarowski, "*Notes on Orthogonal Wavelets and Wavelet Packets*", Report no. 1-95, Department of Electrical and Computer Engineering, Queen's University, Kingston, Ontario, Canada, Nov. 3, 1995.

10. Mike Sablatash, "Multiple Access Communications Based on Wavelet Packet Signal Decomposition and Smart Joint Detection", *1995 Canadian Workshop on Information Theory*, Lac Delage, Quebec, 1995.

11. R. Lindsey, "*Generalized Orthogonally Multiplexed Communication via Wavelet Packet Basis*". Ph.D. Dissertation, Faculty of the Russ College of Engineering and Technology, Ohio University, 1995.

12. R. Lindsey, "Wavelet packet modulation for orthogonally multiplexed communications. *IEEE Trans. Signal Processing*, Vol. 45 , 1997, pp 1336-1339.
13. M. Sablatash, J. H. Lodge, and W. F. McGee, "Equivalence between vestigial sideband (VSB) and offset quadrature phase shift (OQPSK) modulations and relationships to wavelet packet-based multiplexing,". in *Proc. 18th Biennial Symposium on Commun., Queen's University, Kingston, Ontario, Canada*, 1996, pp. 339-342.
14. B. Hirosaki, An orthogonally multiplexed QAM system using the discrete Fourier transform. *IEEE Trans. Commun.* Vol. COM-29, 1981, pp. 982-989.

15. B.Hirosaki, S. Hasegawa, and A. Sabato,. "Advanced groupband data modem using orthogonally multiplexed QAM technique," *IEEE Trans. Commun. Vol.* COM-34 , 1986, pp. 587-592.

16. R. G. Gitlin and E.Y Ho., " The performance of staggered quadrature amplitude modulation in the presence of phase jitter", *IEEE Trans. Commun.*, Vol. COM-23 , 1975, pp. 348-352.

17. M. Sablatash, J. H. Lodge, and W. F. McGee, "The design of filter banks with specified minimum stopband attenuation for wavelet packet-based multiple access communication,". in *Proc. 18th Biennial Symposium on Commun., Queen's University, Kingston, Ontario, Canada*, 1996, pp. 53-56.

18. M. Sablatash, W. F. McGee, and J. Lodge, "Designs of prototype filters for calculation of polyphase components of DFT filter banks and simulation studies to evaluate the bit error rate performance of a multiplexer-demultiplexer filter bank transmitter-receiver for downstream transmission with phase and timing errors at the receiver". In *Proc. of Wireless '97, the 9th Intl. Conf. on Wireless Commun., Coast Plaza Hotel, Calgary, Alberta, Canada*, 1997, Vol. 1, pp. 222-241.

19. M. Sablatash, W. F. McGee, and J. Lodge, "Spectrum-efficient bandwidth-on-demand multi-user transmission based on concatenation of filter bank trees and polyphase filters", in. *Proc. of the 19th Biennial Symposium on Commun., Queen's University, Kingston, Ontario, Canada*, 1998, pp. 382-385.

20. M. Sablatash, W. F. McGee, and J. Lodge, "Effect of carrier phase and symbol timing on the performance of a spectrum multi-carrier transmission system," *Proc. of the 19th Biennial Symposium on Commun., Queen's University, Kingston,, Ontario, Canada*, 1998, pp. 386-390.

21. W. F. McGee, "Variance of Likelihood Estimates for VSB Timing and Channel Phase," *Report*, Project U6800-7-1525/00, 10 Dec 1997.

22. J. Proakis, "*Digital Communications*", 2nd Edition, New York: McGraw-Hill, 1989

23. R. F. Pawula, "A New Formula for MDPSK Symbol Error Probability," *IEEE Communications Letters*, Vol. 2, No. 10, Oct 1998, pp. 271-272.

24. M. K. Simon, "A unified approach to the performance analysis of digital communications over generalized Rayleigh channels," *Proc. IEEE*, Vol. 86, Sept. 1998, pp 1860-1877.

25. M. Schwartz, W. R. Bennett and S. Stein, "*Communication Systems and Techniques*," New York: McGraw-Hill, 1966.

26. G. F. M. Beenker, T. A. C. M. Claasen and P. W. C. Hermens, "Binary Sequences with a Maximally Flat Amplitude Spectrum," *Philips Journal Research*, Vol. 40, 1985, pp. 289-304.

27. W. F. McGee, "Synchronization Sequences for VSB Signals," *Report*, Project U6800-7-1525/00, 10 Dec 1997.

28. W. F. McGee, "Optimum Offset QPSK sequences for Packet Synchronization," *Report*, Project U6800-7-1525/00, 19 Jan 1998.

29. J.R.Ball, "A real-time fading simulator for mobile radio", *The Radio and Electronic Engineer*, Vol.52, No.10, Oct.1982.

Appendixes

Appendix 1: Modeling Rician channels

The channel specified for this simulation is common to work in the mobile satellite area, it is a Rician channel. This means that the received signal is the sum of two signals, in one of which the transmitted signal is received without change or attenuation, and the other in which the transmitted signal has been randomly modulated with slowly changing complex Gaussian noise. The Gaussian noise is specified by its power spectrum, and the average power of its Rayleigh component is $10^{-K/10}$. The spectral shape is obtained by passing white complex Gaussian noise through an IIR filter that matches the spectrum as measured in field measurements. It is possible to store the final conditions of the state variables of the IIR filter, and use them as initial conditions for another simulation run. This filter[29] is, as well, specified by its Doppler bandwidth, which is a measure of the velocity of the antennas and the carrier frequency.

The modulation is performed by multiplying the transmitted complex data signal by a complex sequence on a sample by sample basis. The question is how to obtain this complex sequence.

We found two problems that required some consideration.

When the Doppler shift is small, the Rician component does not change much from symbol to symbol, or even from block to block. In terms of simulation, this means that error rate estimates from block to block are not independent, and therefore to obtain a good set of independent error rate estimates takes more time.

The second problem occurs with the use of the IIR filter. At startup we do not know good initial values of the state variables of the IIR filter. Therefore the output signal starts out small, but exhibits large fluctuations in phase. After the output is in a steady state, the output is larger, but does not exhibit such great phase changes from sample to sample.

Our approach is to initially run the filter for a large number of samples and throw away the output. Between samples, we run the filter for a number of samples and throw away the output.

The routines that are used are CRC-standard MATLAB programs, modified to be able to save final conditions.

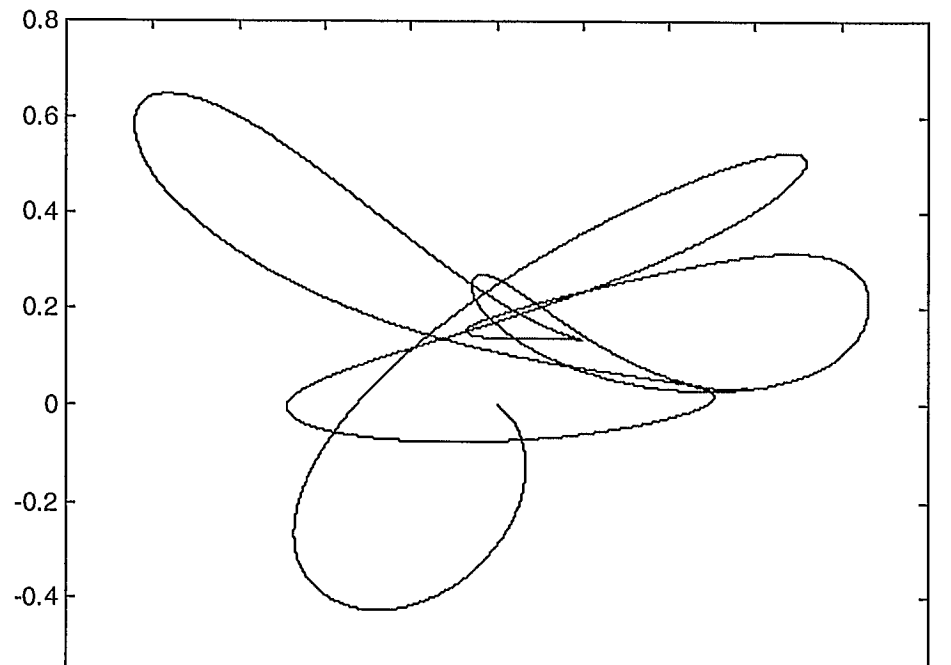


Figure A1.1 Imaginary (y axis) and real (x axis) of 10000 Rician variates K=7, fd=100 Hz, fs=160000 Hz

Appendix 2: BPSK Error rate of Rician channels with perfect phase:theory

In order to compare our results with a perfectly synchronized system, it is useful to know the ideal behavior of a BPSK system in a Rician channel with perfect channel information.[2] In the communication system under study the transmitted signal a(t) is modulated with a complex Gaussian noise process r(t) with mean 1 and variance $10^{-K/10}$. In the receiver complex Gaussian noise is added; the received signal is then a(t)r(t)+n(t). The demodulator forms an estimate of the channel, assumed perfect for now, and then multiplies the received signal by the complex conjugate r*(t) of the channel estimate and the real part of the resulting signal is determined. If we assume that a transmitted 'one' was sent, [a(t)=1] then the error is

$$P_E = \Pr(\mathrm{Re}(|r(t)|^2 + n(t)r^*(t)) < 0) \qquad (5)$$

$$= \Pr(|r(t)|^2 + \frac{1}{2}(r^*(t)n(t) + n^*(t)r(t)) < 0)$$

$$= \Pr\left(\begin{pmatrix} r^*(t) & n^*(t) \end{pmatrix} \begin{pmatrix} 1 & 0.5 \\ 0.5 & 0 \end{pmatrix} \begin{pmatrix} r(t) \\ n(t) \end{pmatrix} < 0 \right)$$

$$E\begin{pmatrix} r(t) \\ n(t) \end{pmatrix} = \begin{pmatrix} \bar{r} \\ \bar{n} \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

$$E\left(\begin{pmatrix} r-\bar{r} \\ n-\bar{n} \end{pmatrix} \begin{pmatrix} r-\bar{r} \\ n-\bar{n} \end{pmatrix}^H \right) = \begin{pmatrix} 10^{-K/10} & 0 \\ 0 & N_0/E_b \end{pmatrix}$$

This is just the probability that a quadratic form in complex Gaussian variates is negative. This is a well-known problem in statistical communications[21,25], and the solution may be given in terms of the Marcum Q-function Q(a,b).

If the signal used for the demodulation is only partially correlated with the channel, then the error rate may be considered as the following probability of a quadratic form in 2 complex Gaussian random variables.

---

[2] Proakis[22] has a detailed discussion in his Appendix C1. An estimate of the channel that is determined from a *pilot signal* has an additional noise that is inversely proportional to the pilot symbol energy. The *clairvoyant estimate* modulates the recovered signal with the recently demodulated data. Also, Proakis considers averaging the estimate over the past with a given weight function with coefficients c_i, which he calls the normalized pilot or clairvoyant estimate. Our estimation method is equivalent to the clairvoyant method of Proakis. The perfect channel estimator has no associated noise. The magnitude is still fluctuating, and this averaging is what produces the different error rates. In the simulation the estimates have fluctuations about the perfect estimate.

$$
\begin{aligned}
P_E &= \Pr\left(\mathrm{Re}\left(c\hat{c}^* + n\hat{c}^*\right) < 0\right) \qquad (6)\\
&= \Pr\left(\mathrm{Re}\left((c+n)\hat{c}^*\right) < 0\right)\\
&= \Pr\left(\begin{pmatrix} (c+n)^* & \hat{c}^* \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} c+n \\ \hat{c} \end{pmatrix} \right)\\
&E(c+n) = 1\\
&E(\hat{c}) = 1\\
&E\left(\begin{pmatrix} c+n-1 \\ \hat{c}-1 \end{pmatrix} \begin{pmatrix} (c+n-1)^* & (\hat{c}-1) \end{pmatrix} \right) = \begin{pmatrix} 10^{-K/10} + E_b/N_0 & 10^{-K/10}\rho(\tau) \\ 10^{-K/10}\rho^*(\tau) & 10^{-K/10} \end{pmatrix}
\end{aligned}
$$

This problem is now phrased as a standard one, and a simple calculation yields the error rate.

Yet another approach, and the one that we prefer, is to express the error rate as an integral in the following steps, and then to approximate the integral.

$$
\begin{aligned}
P_E &= \Pr\left(|\, r(t)\,|^2 + \mathrm{Re}(n(t)r^*(t)) < 0\right) \qquad (7)\\
&= \underset{r}{E}\left(\Pr\left(|\, r(t)\,|^2 + \mathrm{Re}(n(t)r^*(t)) < 0 \mid r(t)\right)\right)\\
&= \underset{r}{E}\left(Q\left(\sqrt{\frac{2E_b}{N_0}}\, |\, r(t)\,| \right) \right)\\
&= \underset{r}{E}\left(\frac{1}{\pi} \int_0^{\pi/2} \exp\left(-\frac{2E_b\,|\, r(t)\,|^2}{2N_0 \sin^2\theta} \right) d\theta \right)\\
&= \frac{1}{\pi} \int_0^{\pi/2} \exp\left(-\frac{E_b}{E_b 10^{-K/10} + N_0 \sin^2\theta} \right) \frac{N_0 \sin^2\theta}{E_b 10^{-K/10} + N_0 \sin^2\theta}\, d\theta
\end{aligned}
$$

We use this formula to determine the Rician channel error rate using the MATLAB routine ricetheory

N.B. In this exposition we are deviating from previous CRC definition of the factor K, in order to bring it in line with current texts. In previous CRC publications, a typical channel had a K factor of –8 dB.

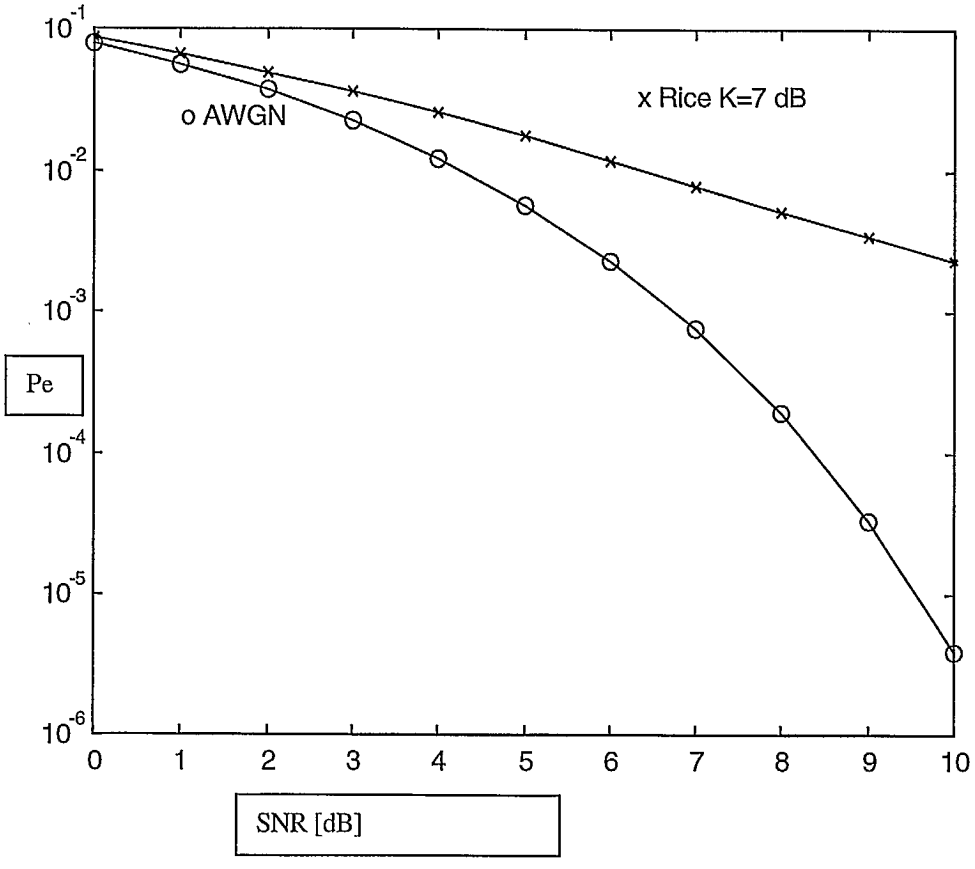


Figure A2.1 Error rate vs. SNR for Rician channel, K=7 dB. Perfect phase synchronization is assumed. The reference line for BPSK is shown, and the Rician channel has the larger error rate..

Appendix 3: Synchronization Algorithm: Statistical properties

The synchronization algorithm consists of taking a sequence of 128 data samples, inserting 32 zero samples in the middle, making 160 samples in all, and passing these samples through the 3-level filterbank tree. At the output, the middle block of 32 samples is selected, and, using a matrix inversion, it is determined what the 32 previously zero samples should be to produce the same samples in the timing window. These now non-zero samples are passed through the filterbank, and the resulting signal is subtracted from the previously determined signal, thus causing a null for 32 samples in the high speed output. A known pseudo random sequence is then inserted in these 32 time slots.

To analyze the performance of this algorithm, let the matrix **G** consist of the pulse response of all pulses that have a pulse response that lies in the timing window. It has as many rows (32 in this case) as the window is wide, and as many columns as necessary to hold all the responses. The matrix **G** satisfies

$$\mathbf{G}\mathbf{G}^T = \mathbf{I} \qquad (8)$$

in the ideal case when the filters are perfect reconstruction, and approximately with approximately perfect reconstruction. This equation is a restatement in matrix form of the conditions for perfect reconstruction for analysis/synthesis. The left-hand side may be deduced by explicitly writing out the transfer function from

an input sequence of length 32 to a suitably delayed sequence of 32 samples after an analysis/synthesis pair; the right-hand side is a unit matrix because of the perfect reconstruction property. This matrix G, and the inputs, are divided into two blocks, one defined by the data **d**, and the other by the samples **s** of the inputs that are to be used to make the window samples vanish. If we use **s** for the vector of samples that we are going to introduce to produce the null samples in the output window, and **d** for the vector of relevant data, then we have **s** defined by

$$\mathbf{0} = \mathbf{G}\begin{pmatrix}\mathbf{s}\\ \mathbf{d}\end{pmatrix} = \begin{pmatrix}\mathbf{A} & \mathbf{D}\end{pmatrix}\begin{pmatrix}\mathbf{s}\\ \mathbf{d}\end{pmatrix} \tag{9}$$

where **A** is square, and **0** is a vector of length the window size; 32 in our simulation. Then the solution for the vector **s**, the non-zero samples, is

$$\begin{aligned}\mathbf{As} + \mathbf{Dd} &= \mathbf{0} \\ \mathbf{s} &= -\mathbf{A}^{-1}\mathbf{Dd}\end{aligned} \tag{10}$$

The ideal solution would be to have the output window vanish while **s** is zero, i.e., no need for any correction. However, with our filters, this does not occur. We use as a figure of merit the trace of the expectation (with respect to the data) of $\mathbf{ss}^T$, which measures the average total energy of the samples **s** used to force the window to zero.

Thus

$$\begin{aligned}&E_s = E\left(\mathbf{s}^T\mathbf{s}\right) = E\left(Tr\left(\mathbf{s}^T\mathbf{s}\right)\right) = E\left(Tr\left(\mathbf{ss}^T\right)\right) = \\ &Tr\left(E\left(\mathbf{A}^{-1}\mathbf{Ddd}^T\mathbf{D}^T\mathbf{A}^{T-1}\right)\right) = Tr\left(\mathbf{A}^{-1}\mathbf{DD}^T\mathbf{A}^{T-1}\right) \\ &= Tr\left(\mathbf{DD}^T(\mathbf{AA}^T)^{-1}\right) = Tr\left((\mathbf{AA}^T)^{-1} - \mathbf{I}\right)\end{aligned} \tag{11}$$

where we have used that knowledge that, since the data elements are independent,

$$E\left(\mathbf{dd}^T\right) = \mathbf{I} \tag{12}$$

Furthermore, Tr(**CB**)=Tr(**BC**), and finally the two matrices **A** and **D** satisfy

$$\mathbf{GG}^T = \begin{pmatrix}\mathbf{A} & \mathbf{D}\end{pmatrix}\begin{pmatrix}\mathbf{A}^T\\ \mathbf{D}^T\end{pmatrix} = \mathbf{AA}^T + \mathbf{DD}^T = \mathbf{I} \tag{13}$$

If we replace the signal in the window with one with unit norm, then Trace($(\mathbf{AA}^T)^{-1}$) represents the total energy, and Trace($(\mathbf{AA}^T)^{-1}$) /length(window) represents the average energy per symbol used for synchronization. For a three-level tree, the excess energy ranges from 0 to 7.5 percent.

The expression for the average of the signal in the window that has to be cancelled is the expression

$$\mathbf{E}_C = E(\mathrm{diag}(\mathbf{Ddd}^T\mathbf{D}^H)) = \mathrm{diag}(\mathbf{DD}^H) = \mathrm{diag}(\mathbf{I} - \mathbf{AA}^T). \tag{14}$$

INVERSE SENSITIVITY

Some authors use the ratio of the maximum absolute eigenvalue to the minimum absolute eigenvalue of a matrix as a measure of the numerical stability of the inversion process. However, this ratio depends on the ordering of the rows of the matrix, and thus does not appear too useful. What would be better is to use the ratio of maximum to minimum of the elements of the diagonal matrix of the SVD decomposition of a matrix. According to the Singular Value Decomposition, any matrix **A** may be represented as

$$\mathbf{A} = \mathbf{U}^{\mathbf{H}}\Sigma\mathbf{V}$$

where **U** and **V** are unitary and Σ is a diagonal matrix whose elements are the positive square roots of the eigenvalues of the matrix $\mathbf{A}^{\mathbf{H}}\mathbf{A}$ or $\mathbf{A}\mathbf{A}^{\mathbf{H}}$.

RESULTS

A calculation of the relevant matrices for s filterbank with linear phase filters at each level gives the following results. There are 26 configurations. We plot the ratio of the maximum to minimum eigenvalue of the matrix AA^T for each of the 26 bandwidth-on-demand configurations in Figure, the total excess power per sample in Figure, and all the diagonal elements of $(AA^T)^{-1}$.

CONCLUSION

The use of linear phase filters appears to carry a power penalty of at most 0.3 dB per synchronization sample. However, the energy is not uniform across the timing window.

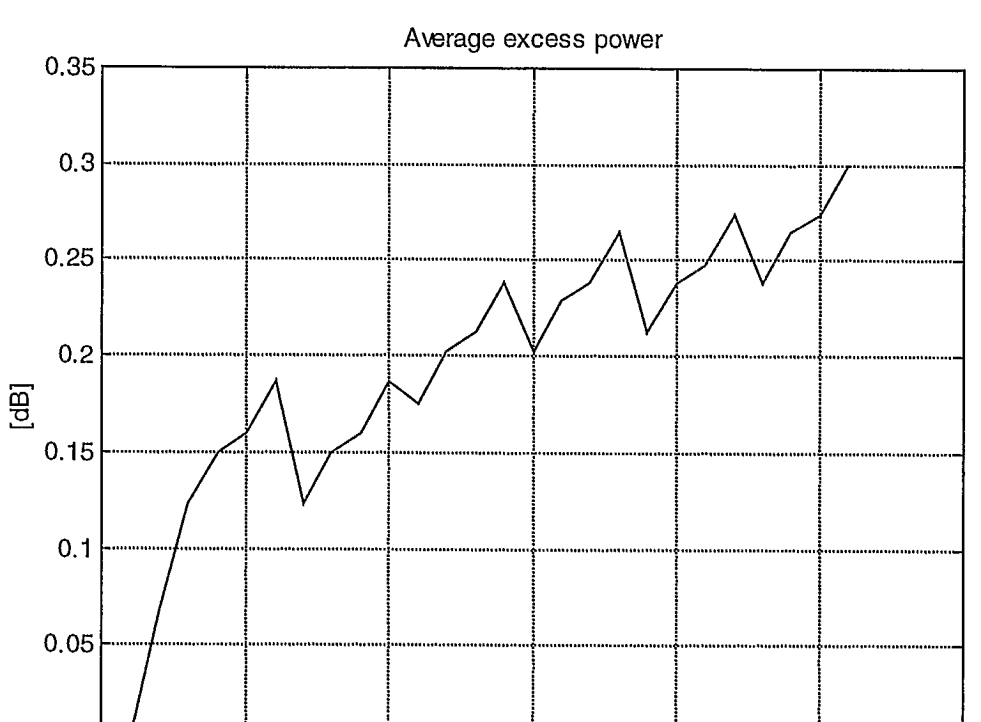


Figure A3.1 Plot of maximum excess power per pilot sample for various configurations. This is averaged over the window. A more appropriate scaling would average over the 160 data samples; thus the impact is about 0.05 dB per bit, typically (0.25*32/160).

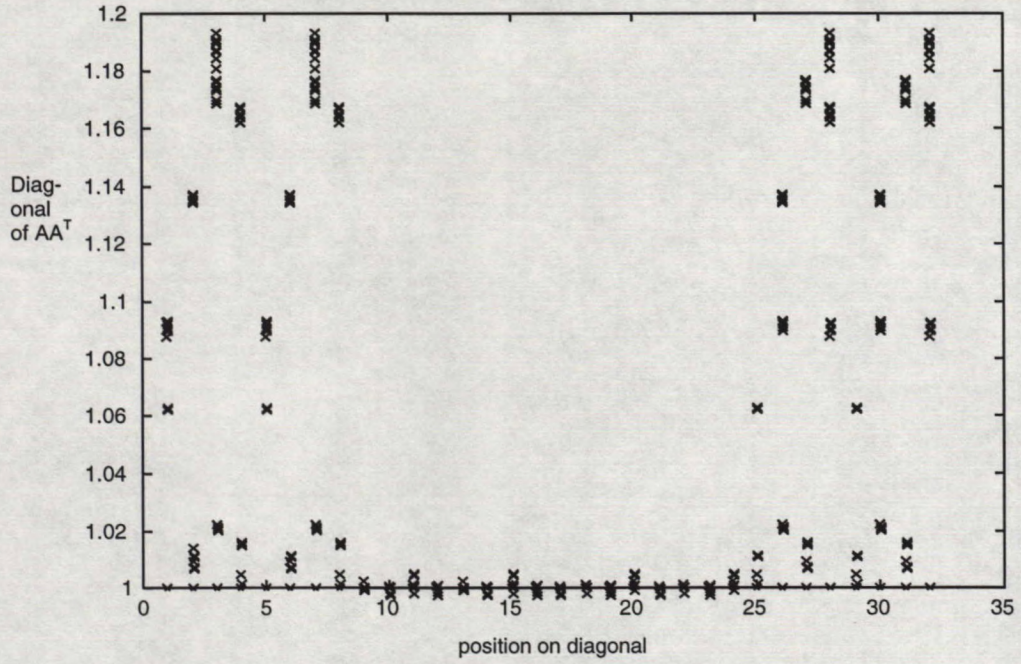


Figure A3.2 Plot of diagonal elements of inv(AA^T) for all configurations. The spread is greatest near the ends of the timing window.

Appendix 4: Qualitative Explanation of Influence of Doppler on Error Rate

The system data rate is 160 kb/s, and this is composed of as many as 8 channels at 1/8 the rate, i.e. 20 kb/s. The system is VSB, which means that the signals pass through receive filters with bandwidths that are as little as 10 kHz, and at most 80 kHz.

We send a block of 64 bits, then a timing segment of 32 bits, and a data block of another 64 bits. The phase is estimated at the center of the block using the middle 32 bits. Thus, a sampled bit may be as much as 80 bits away from the center.

For this analysis we assume that the phase estimate is correct. Therefore we multiply the received signal, 1+z+n, where z is the Rayleigh component of Rician fading, and n is Gaussian noise, by a unit magnitude constant whose phase cancels that of the signal phase at the center of the block. Since n is uncorrelated from sample to sample, the error rate is then the probability

$$P_E(i) = \Pr\left(\mathrm{Re}\left((1+z(i)) \frac{1+z^*(0)}{|1+z^*(0)|} \right) + \mathrm{Re}(n) < 0 \right)$$

If the phase estimate were constant across the 80 bits, then the error rate should fall on the theoretical curves for Rician fading channels. But this is not the case, and therefore consider the difference between the Rayleigh component at time i and time 0, $\delta z(i) = z(i) - z(0)$.

Then the error probability with all 1s is

$$P_E(i) = \Pr\left(|1+z(0)| + \mathrm{Re}\left(\delta z(i) \frac{1+z^*(0)}{|1+z^*(0)|} \right) + \mathrm{Re}(n) < 0 \right)$$

This is the sum of a constant and two noise terms, one of which is the thermal noise, and one of which is due to the Rayleigh component of the Rician channel. The power of the Rayleigh component is'

$$\begin{aligned} S^2 &= E\left(\mathrm{Re}\left(\left(\delta z(i) \frac{1+z^*(0)}{|1+z^*(0)|} \right) \mathrm{Re}\left(\delta z(i) \frac{1+z^*(0)}{|1+z^*(0)|} \right) \right) \right) \\ &= \frac{1}{2} E(|\delta z(i)|^2) + E\left(\mathrm{Re}\left(\delta z(i) \frac{1+z^*(0)}{|1+z^*(0)|} \right)^2 \right) \\ &\cong \frac{1}{2} E(|\delta z(i)|^2) \end{aligned}$$

ignoring the second term.

Then S^2 may be shown to be the difference of the autocorrelation function of the Rayleigh component at time i and time 0. In particular, it vanishes when i is zero. That is

$$S(i)^2 = R_z(0) - R_z(i) > 0$$

This formula expresses the extra power in terms of the autocorrelation function of the Doppler channel passed through the channel filter. When twice the Doppler frequency f_D is less than the receive filter bandwidth, then the autocorrelation function is just the Fourier Transform of the spectrum of the Doppler shifted Rayleigh component, and will not depend on the channel bandwidth.

When the Doppler frequency is much larger than the receiver filter bandwidth, then the autocorrelation function approaches that of the channel filter, but with a reduced magnitude because the fraction of the Doppler power in band is reduced, inversely proportional to the Doppler shift.

In between there is a broad maximum as the peak of the autocorrelation function moves across the block, making some bits in the block more error prone than others.

Consequently, we would expect, based on this argument, that the error rate curve will show a response which is equivalent to extra noise. The noise power is proportional to the Doppler frequency for small Doppler frequencies, approaches 1.4 times the power over a broad range, and falls off with the inverse of the Doppler frequency for very large Doppler frequencies. The factor 1.4 is due to the ringing of the autocorrelation function, a sin(x)/x phenomenon.

For the various channel bandwidths, we would expect that the Doppler causing the peak error rate will be approached from the low side the same for all the channels, but will start to drop off first for the lower bandwidth channels.

In conclusion, the following appears to explain the behavior. When a phase estimate is used, those symbols further away have an additional noise term which is proportional to difference in the channel between the two times. The squared magnitude of this added noise is the Fourier transform of the power spectrum of the Doppler Rayleigh component after passing through the channel filter, less the dc value, according to equation X. For a particular position, as the Doppler increases, this noise will increase at first proportional to the square of the Doppler, and then decrease as the inverse of the Doppler, as the channel filter comes into play.

Appendix 5: Good sequences for timing and phase estimation

The synchronization method that is analyzed in this report consists of creating a dead zone in a sequence, and injecting a pilot sequence into the dead zone. The timing and phase synchronization is achieved at the receiver by investigating the output of a filter matched to the inserted pilot sequence.

In synchronization there are two modes of operation, a startup mode, and a steady state mode. In the startup mode the receiver has no idea where the pilot sequence is, in time. Therefore, the matched filter output must be scanned continuously, and a peak found. Once found, the channel phase and gain may be determined, the symbol timing determined, and, in time, the frequency offset determined. We assume that the transmitter does not know which mode a receiver is in, and that the pilot sequence does not change. A good sequence for this purpose would be one in which the peak of the matched filter is likely to be found in the presence of data and noise. Also we have the intuitive idea that the spectrum of the pilot sequence should be relatively flat, thereby probing the whole bandwidth.

In the field of sequence design there have been many advances, particularly in the early days of radar, where the sequence design is fundamental. These have been summarized in a previous report [27]. If the sequence is constrained to be real, but not otherwise, then there are design techniques which produce autocorrelation functions that are zero except at the end points. If complex, there are sequences based on quadratic residues that have excellent autocorrelation properties. However, we have decided to restrict our attention to real, binary (+1 or –1) pilot sequences, since these are more consistent with VSB signaling.

In the steady state mode the receiver knows where the pilot sequence is in time, and the only question is the sample value of the matched filter at this time; this gives the channel amplitude and phase. In terms of this channel amplitude and phase estimation, all binary sequences of a given length are equivalent, since the noise reduction produced by the matched filter is only dependent of the sum of squares of the filter coefficients, which, for +1 and –1 coefficients, is a constant independent of the sign of the coefficients.

Symbol timing is a bit more complicated. Typically the timing recovery is approached by evaluating the matched filter output by oversampling by a factor of 2, and taking the phase-determining sample, and the adjacent samples, to predict where the magnitude of the autocorrelation function would have had its peak. In order to disassociate the phase and timing estimation, it is preferable to operate on an offset QPSK signal at baseband. [

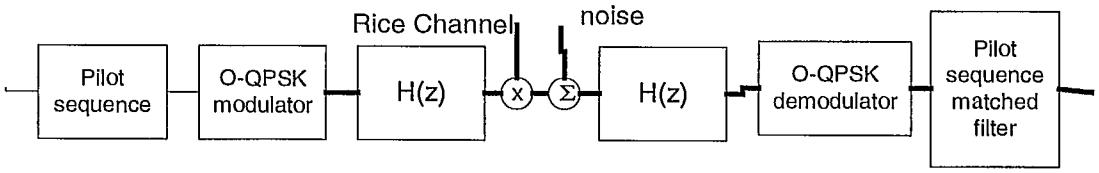


Figure A5.1 Model of the pilot sequence inserting, OQPSK modulation, filtering, channel gain and phase multiplication, noise injection, and filtering at the receiver. The output is a complex sequence.

The pulse sequence used in the simulation has very good properties for synchronization, and has the autocorrelation function shown in Figure A5.2

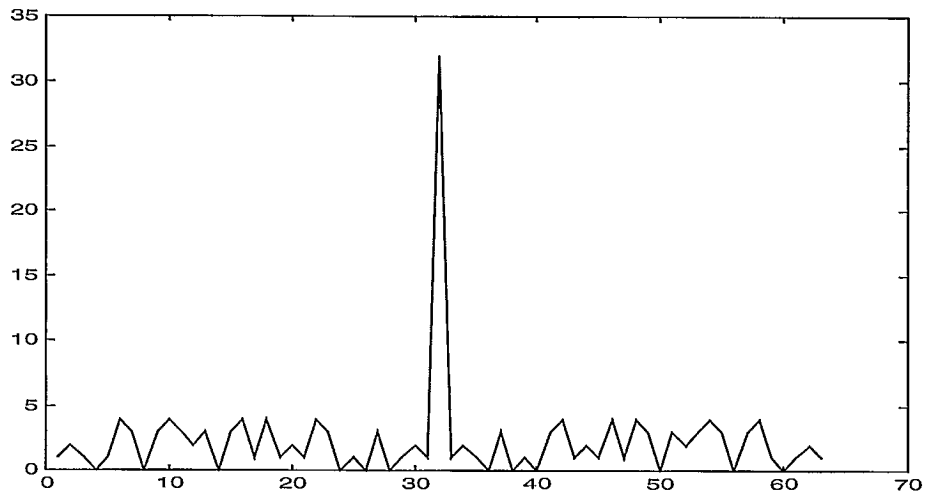


Figure A5.2Autocorrelation function of sequence used in simulation.

Although this sequence is very good, it is probably not optimum. The difficulty here is in determine optimality. We have not determined such an optimal sequence, but record here some other optimal sequences.

Golay has studied such sequences, and considered a class of sequences of odd length which have autocorrelation functions vanishing at the odd sample times. In particular, he considers the class of skew-symmetric binary sequences. These have the following form. Given any two binary sequences, A and B, of the appropriate length, form the sequences [A c fliplr(A)] and [B –fliplr(B)], where c is arbitrary and fliplr reverses the coefficients, and interleave the two sequences to form the desired sequence. This sequence, by design, will have vanishing odd coefficients for its autocorrelation sequence. He chooses from among different sequences by choosing those with the best Merit Factor M defined by

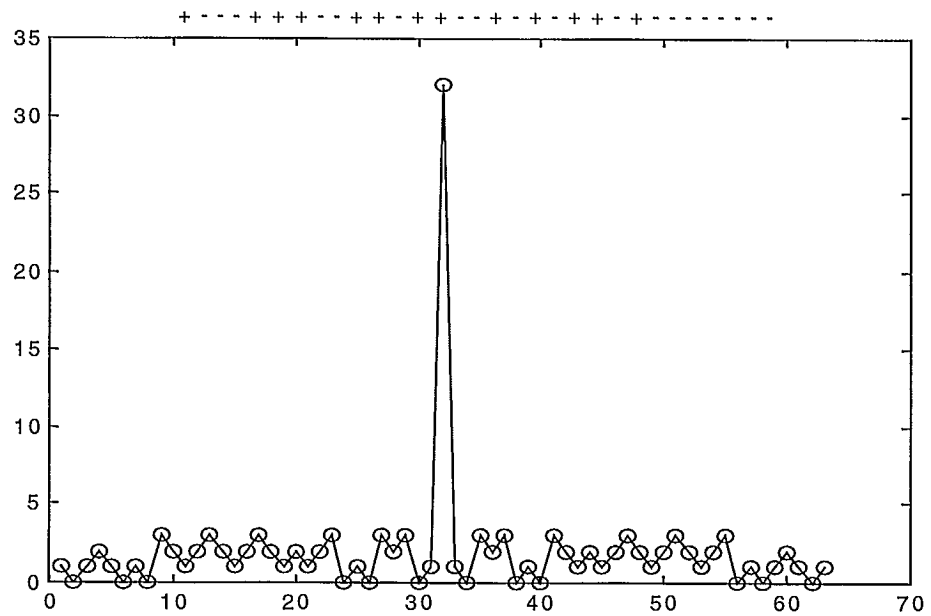
$$M = \frac{r^2_{peak}}{\sum_{i \neq peak} r_i^2} \qquad \text{(A5.15)}$$

Where r_k is an element of the autocorrelation function. The higher the figure of merit, the less the deviation of the spectrum from a constant. This follows, because from Parseval's theorem

$$M = \frac{\left|\int S(f)df\right|^2}{\int \left|S(f) - \int S(f)df\right|^2 df} \qquad \text{(A5.16)}$$

The other merit factor P is

$$P = \frac{|r_{peak}|}{\max_{i \neq peak} |r_i|} \qquad \text{(A5.17)}$$



One approach is to find the length-32 random sequence with the smallest peak of the autocorrelation, thus minimizing P of Equation A5.3. This has peak 3, and one such sequence is shown with its autocorrelation function in Figure A5.3.

Figure A5.3 Absolute value of autocorrelation function of indicated sequence. This is a sequence of length 32 with a minimum of the maximum of the autocorrelation.

Another sequence would minimize A5.2, figure of merit M. One such sequence is exhibited in Figure A5.4.

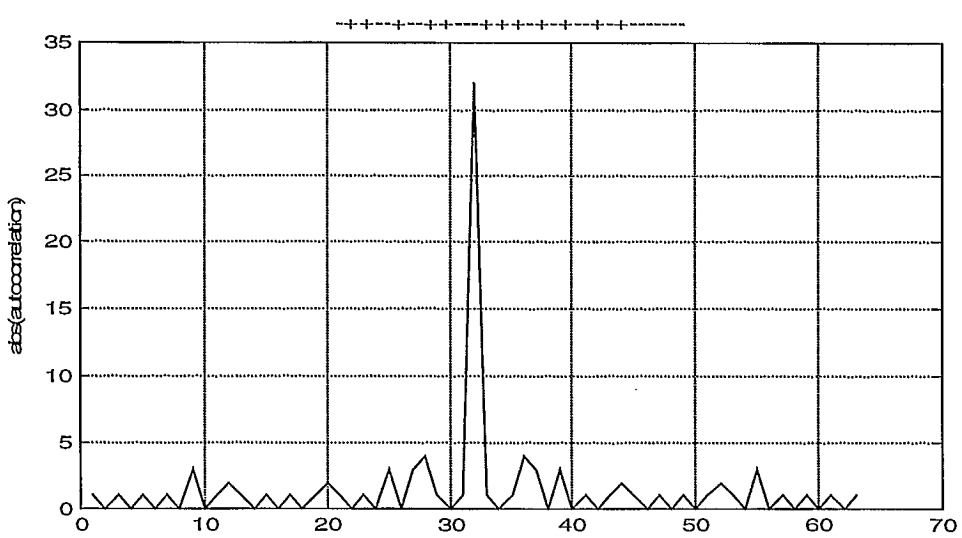


Figure A5.4 Absolute value of autocorrelation function of sequence of length 32 and with sum of squares as small as possible (sum of squares=128 off center).

An optimal 32-bit sequence for synchronization remains open, but one of these three sequences is considered very good.

Appendix 6: Filters used in the study

Here are the coefficients of the filters used in this study. They are linear phase, half-band filters, with 40 dB stopband loss, and the equiripple loss characteristic.

Table 5 Filter Coefficients

Coefficient #/filter	f1	f2	f3
Excess BW (%)	50	25	12.5
Delay (samples)	5.5	10.5	22.5
1	-6.87282356e-003	3.87848742e-003	-1.02456120e-003
2	3.37124935e-002	-1.02390133e-002	-4.13671411e-003
3	-8.09744528e-003	-1.80315756e-003	7.11335703e-003
4	-1.24972246e-001	2.12510093e-002	5.05335979e-004
5	1.32792120e-001	-2.62846824e-003	-7.82692052e-003
6	6.82274764e-001	-4.06077573e-002	-4.80084060e-004
7		1.63426539e-002	1.13895673e-002
8		7.29261879e-002	-7.25911646e-004
9		-5.11113258e-002	-1.59073688e-002
10		-1.41188532e-001	3.01130592e-003
11		1.78057976e-001	2.17203829e-002
12		6.61764943e-001	-6.70255334e-003
13			-2.91532383e-002
14			1.26123575e-002
15			3.91553002e-002
16			-2.20724717e-002
17			-5.36415865e-002
18			3.85816626e-002
19			7.84410425e-002
20			-7.31798156e-002
21			-1.37926847e-001
22			1.96470873e-001
23			6.49947882e-001

Figure Coefficients, delay, and excess bandwidth of the filters used in this study. The filters are symmetrical, and only half the coefficients are tabulated.

The frequency response of the filters is shown in the accompanying figures.

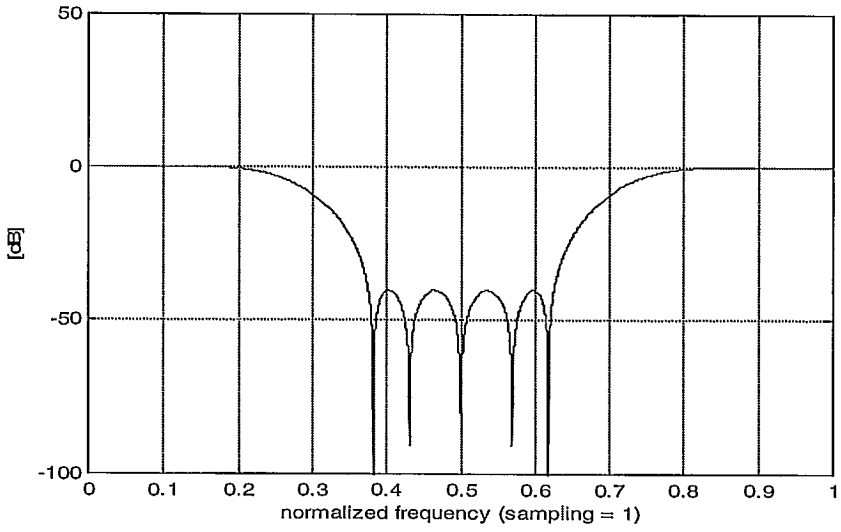


Figure A6.1 Attenuation of filter f1. The stopband starts at 3/8=0.375.

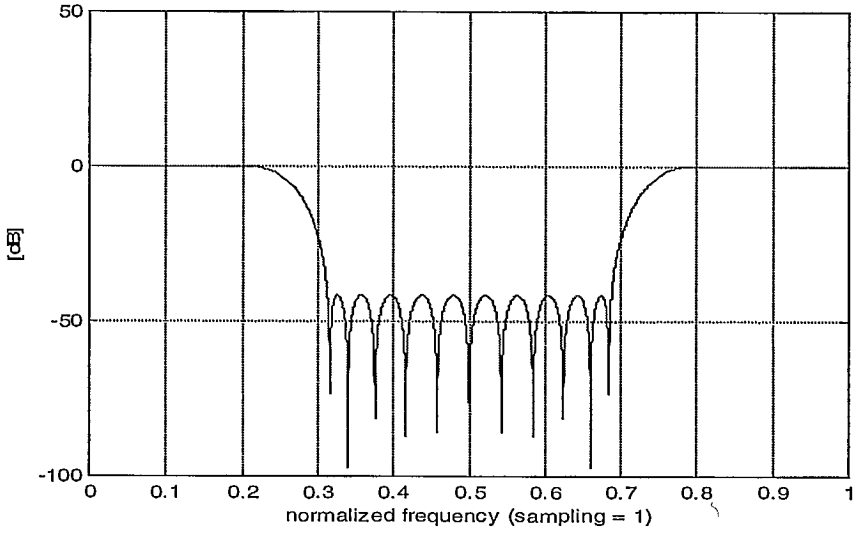


Figure A6.2 Attenuation of filter f2. The stopband starts at 5/16=0.3125.

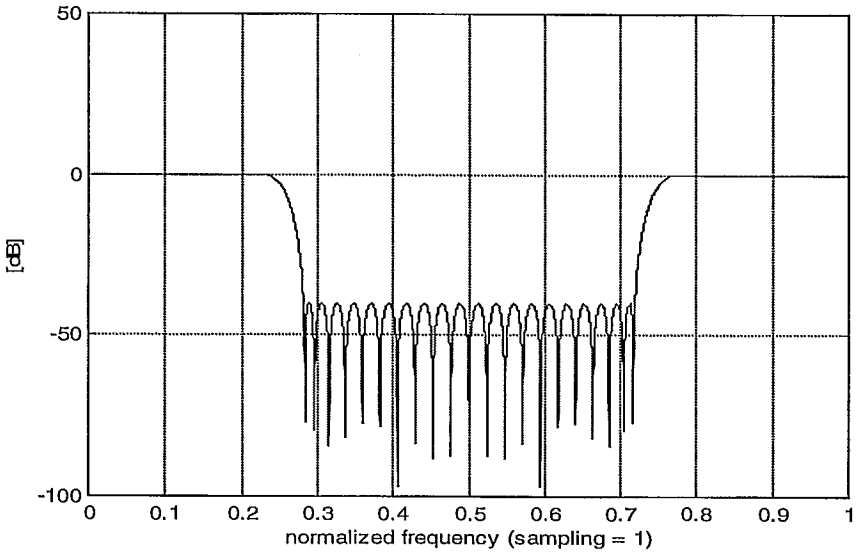


Figure A6.3 Attenuation of filter f3. The stopband starts at 9/32=0.28125.

MATLAB source

contents.m

```
% rsim3 Rician channel simulator
% W F McGee
% contract U6800-9-0526
% 31 Jan 1999
% User settable variables
% dointerp : boolean  true, 3 point interpolation, false, 1 point interpolation
% delta : affects number of simualtion runs choose 0.03 generally, smaller
%                          delta, more runs
% validindex number between 1 and 26 to select configuration
% Doppler: row matrix of Doppler frequencies
% signaling : signaling rate in symbols per second
% RiceFactor : used to describe the power in the Rayleigh component of the Rice
%                         channel
% mynsr : row matrix of noise to signal ratios in power ratios [i.e. not dB!]
% dlen: length of data block per channel
% numchan : number of channels
% slen: length of pilot sequence per channel
% f1, f2, f3: filter coefficients as .mat files
% valid : file of valid configuration numbers
%
% Output1: plot of bit error rate (log scale) vs SNR [dB] for various Doppler
% shifts. Also saved as file 'testrun'
% savephase variance of phase for various Doppler, SNR

Requires the following routines
%    q: normal probability
%    plotpe: plot error rate vs SNR
%    maketree: generates filters needed for simulation tree
%       high: makes high pass filter from lowpass
%       f1.mat: filter coefficients (c) W F McGee 1998
%       f2.mat:      "
%       f3.mat:      "
%    valid.mat: definition of valid tree configurations
%    makeAmatrix: generates the A matrix used in addseq
%    lmffilt1: generates an IIR filter coefficients for channel simulation
%       NoiseBW: used to calibrate IIR filter
%    datgen: generates random data
%    nrequired: the number of error events required for pretty curves
%    rice3: generates a vector of Rician channel values
%    addseq: generates random data, filters it, measures residue,
%                            inserts data to cancel, inserts mseq
%       lintree: does the tree signal processing
%    makesqam: transforms a sequence into an offset QPSK sequence
%    chancgn: Gaussian noise generator
%    getphase4: determines channel estimates from matched filter
%    myinterp4: interpolates a channel estimate
%    untree: analysis tree
%       decode: recovers the input data estimate
%    ricetheory: theoretical error rate for a Rician channel, perfect sync
```

rsim3.m

```
% rsim3
% uses Lagrange interpolation
% simulation using linear phase filters
% filters are f1,f2,f3
% W F McGee
% contract U6800-9-0526
% 31 Jan 1999
% includes OQAM
hold off
clear all   %Clear the memory.
debug=(1==0)
dointerp=(1==1)
delta=0.03;      % for the number of trials 0.1 for rough work 0.01 for
publications
dot='ox+*sd^v<>ph'; % for plotting

validindex=26;
Doppler=[ 100.0,  300.0, 1000.0, 3000.0, 10000.0, 30000.0];
Doppler=[100.0,200.0,300.0,400.0,500.0];
Doppler=[100.0];
signaling=160000;
randn('state',sum(100*clock));  %randomize the random number generator
RiceFactor=7;
format short e
format compact

titl=['BER vs Eb/N0 [dB] Rice k=' int2str(RiceFactor) ' dB.'];
titl=[titl ' Signaling ' int2str(signaling) ' b/s. '];
if dointerp
   titl=[titl ' Interpolation. Configuration ']
else
   titl=[titl ' No Interpolation. Configuration ']
end

mynsr=[0.2 0.4 0.6 0.8 1.0];
if debug
   mynsr(1)=0.1;
end

mseq=[-1 -1 -1 1 -1 -1 1 -1 1 1 -1 -1 1 1 1 1 1 -1 -1 -1 1 1 -1 1 1 1 -1 1 -1 1
-1 -1]';

snrs=zeros(size(mynsr));
for i=1:length(snrs);
   snrs(i)=-10*log10(mynsr(i));
end
snrs

% plot the curve for no doppler, no Rayleigh component (K=infinity)
rate=q(sqrt(2*10.^(snrs/10)));
plotpe(snrs,rate,' ');

% filterbank parameters
dlen=16;    % the number of bits per data block per channel.
numchan=8; % number of input channels
slen=4;     % synchronization length
```

```
blen=dlen+slen;
spacing=numchan*blen;
npint=3; % number of phase interpolations

maketree % calculates the filters required for the tree
         % loads f1 and f2, f3, declares hf1, hf2 and hf3
         % t1 ht1 t2 ht2 t3 and ht3
load valid; % the 26 valid configurations
configuration=valid(validindex);  %pick one
titl=[titl int2str(configuration)]
is1a=(bitand(configuration,1)==1);
is1b=(bitand(configuration,2)==2);
is1c=(bitand(configuration,4)==4);
is1d=(bitand(configuration,8)==8);
is2a=(bitand(configuration,16)==16);
is2b=(bitand(configuration,32)==32);
is3a=(bitand(configuration,64)==64);
p=['-- FR'];
if debug % plot the configuration
  ['--' p((2*is1a+1):(2*is1a+2)) '--v']
  ['      |--' p((2*is2a+1):(2*is2a+2)) '--v']
  ['--' p((2*is1b+1):(2*is1b+2)) '--^']
  ['             |--' p((2*is3a+1):(2*is3a+2)) '--']
  ['--' p((2*is1c+1):(2*is1c+2)) '--v']
  ['      |--' p((2*is2b+1):(2*is2b+2)) '--^']
  ['--' p((2*is1d+1):(2*is1d+2)) '--^']
end

Astart=4*floor((length(f1)-1)/2)...
   +2*floor((length(f2)-1)/2)...
   +floor((length(f3)-1)/2) +1 ; % the beginning of the window
start=Astart+numchan*floor(dlen/2); % the beginning of the window

makeAmatrix; % calculate the A matrix

offset=4*(length(f1)-1)+2*(length(f2)-1)+2*(length(f3)-1)+1; % includes the VSB
filters

esave=zeros(32,1);
extra=0;
nsave=0;
savephase=zeros(length(Doppler),length(snrs));

% the calculation is repreated for every Doppler shift and every SNR
for ps=1:length(Doppler)
   Doppler(ps) % print out the Doppler
   % initial the IIR filter
   zinitial=[0];
   [top,bottom]=lmffilt1(Doppler(ps)/signaling);
   [junk,zinitial]=rice3(10000,RiceFactor,zinitial,top,bottom);


   rate=[];
      for s=1:length(snrs)
      meantime=0;
      vartime=0;
      varphase=0;
```

```
      montecarlo=0;      %Number of MONTE CARLO trials
            esno(s)=snrs(s);
            numerr= 0;  %At start of run set error count to zero.
      while numerr<nrequired(delta)
         % pitch away some Rician samples
         [junk,zinitial]=rice3(10000,RiceFactor,zinitial,top,bottom);
         clear junk; % why waste memory
         montecarlo=montecarlo+1;
         for ph=1:npint
            addseq % generate random data, put into tree, calculate window stuff
                   % insert data to remove the window stuff
                   % stick in the magic sequence
            sigtot2d(:,ph)=sigtot;
            datablsave3d(:,:,ph)=datablsave;
         end
         sigtot=[sigtot2d(:,1); zeros((npint-1)*spacing,1)];

         % add all the stuff together
         if npint>1
            for ph=2:npint
              sigtotlen=length(sigtot2d(:,ph));
              sigtot((1+(ph-1)*spacing):((ph-1)*spacing+sigtotlen))...
                 =sigtot((1+(ph-1)*spacing):((ph-
1)*spacing+sigtotlen))+sigtot2d(:,ph);
            end
         end

         savesig=sigtot;

         sigtot=makesqam(sigtot,j); % generate an offset QPSK signal
         sigtot=conv(sigtot,f3);    % filter it

                        % multiply with the Rician noise
         clear r;
         [r,zinitial]=rice3(length(sigtot),RiceFactor,zinitial,top,bottom) ;
         sigtot=sigtot.*r;
         sigtotal = chancgn(sigtot,esno(s));    % add WGN.

         % RECEIVER

         sigtotal=conv(sigtotal,f3);
         sigtotal=makesqam(sigtotal,-j); % undo the offset QPSK
         chanmod=conv(r,f3);
         recstart=start+(length(f3)-1); % include the VSB filter
         % synchronization
         phaseoffset=recstart+numchan*slen-1;

[chanest1,chanest2,chanest3,t1,t2,t3]=getphase4(sigtotal,mseq,phaseoffset,spacin
g) ;
         t2;

         meantime=((montecarlo-1)*meantime/montecarlo)+(t2/montecarlo);
         if montecarlo>1
            vartime=((montecarlo-2)*vartime/(montecarlo-
1))+montecarlo*(abs(meantime-t2)/(montecarlo-1))^2;
         end
```

```
        chanest=myinterp4(chanest1,chanest2,chanest3,phaseoffset-
floor(length(mseq)/2),spacing,length(sigtotal)).';
        if ~dointerp
           chanest=chanest2;
        end
        % decode the data
        varphase=varphase+rmsphase(chanest,chanmod,80)^2;
        sigtotal=sigtotal.*conj(chanest);
        savesigtotal=sigtotal./abs(chanest);
        savesigtotal=savesigtotal./abs(chanest);
        savesigtotal=conv(savesigtotal,flipud(mseq));
        % hold off
        % plot(savesigtotal)
        % axis([-50 50 -50 50]);
        % drawnow
        sigtotal=real(sigtotal);
        % remove the m sequence
        for i=1:length(mseq)
           sigtotal(recstart+i-1)=0;
           sigtotal(recstart+i+spacing-1)=0;
                       sigtotal(recstart+i+2*spacing-1)=0;
        end

        datastart=4*(length(f1)-1)+2*(length(f2)-1)+2*(length(f3)-1)+1;
        leftover=datastart-numchan*floor(datastart/numchan);
        datastart=floor(datastart/numchan);
        if leftover~=0
           datastart=datastart+1;
           sigtotal=[zeros(numchan-leftover,1);sigtotal];
        end
        % recover the data from the tree
        untree; % script file returns data(sigtotal), clobbers sigtotal
        datastart=datastart+blen;
        guess=[data(datastart:(datastart+floor(dlen/2)-
1),:);data((datastart+floor(dlen/2)+slen):(datastart+dlen+slen-1),:)];
        % count the errors
        numerr=numerr+sum(sum(abs(decode(guess)-datablsave3d(:,:,2))))/2;
        %
fprintf('%5.0f/%5.0f\n',numerr,required(delta,numerr,dlen*montecarlo*numchan))
     end
     savephase(ps,s)=sqrt(varphase/montecarlo)
     montecarlo
     rate(s)=numerr/(montecarlo*dlen*numchan)  %Compute the error rate.
     end
  plotpe(snrs,rate,dot(ps))
  logo=[dot(ps) ':' int2str(Doppler(ps)) ' Hz'];
  t=text(0.4,0.05*10^(-0.2*ps),logo);
  set(t,'FontSize',10);
end % ps
rate=[];
% stick in the theoretical curve for perfect synchronization
for i=1:length(snrs)
  rate=[rate ricetheory(snrs(i),RiceFactor)];
end
plotpe(snrs,rate,' ');
title(titl);
grid
```

```
xlabel('SNR [dB]');
ylabel('Bit Error probability Pe');
hold off
print -dmeta testrun

if nsave>0
   esave=esave/nsave;
   extra=extra/nsave;
end
%plot(esave);
%hold on
%plot(esave,'x');
%plot(1-diag(A*A'),'o');
%plot(1-diag(A*A'));
%titl=['Average Window signal x theory o. Extra energy/160: '
num2str(10*log10(1+extra/160)) ' dB'];
%title(titl);
%hold off
%print -dmeta esave
return
```

q.m

```
function [result]=q(x)
% W F McGee
% contract U6800-9-0526
% 31 Jan 1999
result=0.5*erfc(x/sqrt(2));
```

maketree.m

```
%maketree
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999
load f1;
load f2;
load f3;
% make then column vectors
temp=size(f1);
if temp(1)<temp(2)
   f1=f1';
   f2=f2';
   f3=f3';
end
hf1=high(f1);
hf2=high(f2);
hf3=high(f3);
temp=floor(length(f1)/2);
t1=[zeros(temp-1,1) ;1 ;zeros(temp,1)];
ht1=[zeros(temp,1) ;1;zeros(temp-1,1)];
delay1=temp;
temp=floor(length(f2)/2);
t2=[zeros(temp-1,1) ;1 ;zeros(temp,1)];
ht2=[zeros(temp,1); 1; zeros(temp-1,1)];
delay2=temp;
temp=floor(length(f3)/2);
t3=[zeros(temp-1,1) ;1 ;zeros(temp,1)];
ht3=[zeros(temp,1) ;1 ;zeros(temp-1,1)];
delay3=temp;
```

high.m

```
function a=high(b)
for i=2:2:length(b)
   b(i)=-b(i);
end
a=b;
```

makeAmatrix.m

```
% makes the A matrix
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999
A=[];
numchan
for i=1:numchan
   databl=zeros(1,numchan);
   databl=[databl;databl];
   databl(1,i)=1;
   databl;
   lintree;
   for k=0:(slen-1)
      A=[A sig((Astart-k*8):(Astart-k*8+31))];
   end
end
```

lintree.m

```
% lintree
% calculate the tree output using upfirdn
% input is databl
% output is signal
% brute force
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

if is1a
   sa=upfirdn(databl(:,1),f1,2,1)+upfirdn(databl(:,2),hf1,2,1);
else
   sa=upfirdn(databl(:,1),t1,2,1)+upfirdn(databl(:,2),ht1,2,1);
end

if is1b
   sb=upfirdn(databl(:,3),hf1,2,1)+upfirdn(databl(:,4),f1,2,1);
else
   sb=upfirdn(databl(:,3),ht1,2,1)+upfirdn(databl(:,4),t1,2,1);
end
if is1c
   sc=upfirdn(databl(:,5),f1,2,1)+upfirdn(databl(:,6),hf1,2,1);
else
   sc=upfirdn(databl(:,5),t1,2,1)+upfirdn(databl(:,6),ht1,2,1);
end
if is1d
   sd=upfirdn(databl(:,7),hf1,2,1)+upfirdn(databl(:,8),f1,2,1);
else
   sd=upfirdn(databl(:,7),ht1,2,1)+upfirdn(databl(:,8),t1,2,1);
end
if is2a
   sa=upfirdn(sa,f2,2,1)+upfirdn(sb,hf2,2,1);
else
   sa=upfirdn(sa,t2,2,1)+upfirdn(sb,ht2,2,1);
end

if is2b
   sb=upfirdn(sc,hf2,2,1)+upfirdn(sd,f2,2,1);
else
   sb=upfirdn(sc,ht2,2,1)+upfirdn(sd,t2,2,1);
end
if is3a
   sig=upfirdn(sa,f3,2,1)+upfirdn(sb,hf3,2,1);
else
   sig=upfirdn(sa,t3,2,1)+upfirdn(sb,ht3,2,1);
end
```

lmffilt1.m

```
function [b,a] = lmffilt1(fd)

% lmffilt1 - Computes an IIR filter which will produce the land mobile
%            fading spectrum by taking a bilinear transform of the s-domain
%            transfer function given in the reference.
%            Normalization is built in.
% usage:    [b,a] = lmffilt(fd)
%
% in:       fd - the fading rate relative to the channel sampling rate
%
% out:      b  - the IIR filter feedforward coefficients
%           a  - the IIR filter feedback coefficients
%
% refs:     J.R.Ball, "A real-time fading simulator for mobile radio",
%           The Radio and Electronic Engineer, Vol.52, No.10, Oct.1982.

% Author:  R.J.Young, Feb 91

 % modified
% W F McGee
% contract U6800-9-0526
% 31 Jan 1999
as=[2.6906 3.7805 6.4829 4.9383 3.0950 1.0000];
[b,a]=bilinear(1,as,(1/fd)/(2*pi),1/(2*pi));
power=ceil(log(1/fd)/log(2));
power=2^(power+4);
if power<1024
   power=1024;
end;
mpower=2*noiseBW(b,a,power,0);
b=b/sqrt(mpower);
```

noisebw.m

```
function bw = noisebw(b,a,N,plt)

% noisebw - Computes the equivalent noise BW of a digital filter.
%
% usage:  bw = noisebw(b,a);
%         bw = noisebw(b,a,N);
%         bw = noisebw(b,a,N,plt);
%
% in:     b,a - the taps of the  digital filter
%         N   - the number of frequency points used for the calculation.
%               [default = 1024]
%         plt - 0 to suppress plot, 1 for plot. [default = 1]
%
% out:    bw  - the single sided noise bandwidth of the filter normalized to
%               the sampling rate

% Author: N.Shah, Aug 90

    if (nargin < 3)
        N = 1024;
    end
    if (nargin < 4)
        plt = 1;
    end

    [h,w] = freqz(b,a,N,'whole');
    P = h(:).*conj(h(:));
    bw = sum(P)/(N*P(1)*2);

    if plt == 1
      equiv = 0.0000000001*ones(N,1);
      bins = 2*fix(N*bw/2)+1;
      equiv(N/2+1-bins:N/2+1+bins) = P(1)*ones(2*bins+1,1);
      f = -0.5:1/N:0.5-1/N;
      plot(f,[10*log10(abs(fftshift(P))) 10*log10(equiv)])
      title('Equivalent noise BW')
      xlabel('Normalized frequncy [f/fs]')
      ylabel('Response [dB]')
    end
```

datgen.m

```
function y=datgen(blen,nchan)
%
% datgen returns a blen-by-nchan matrix containing
% random binary data, with each element being either
% a -1 or +1.
%
sp1=rand(blen,nchan);
y=(2*round(sp1))-1;
```

nrequired.m

```
function a=nrequired(delta)
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999
a=floor(1/(delta*delta));
if a<2
   a=2
end
```

rice3.m

```
function [r,zf]=rice3(N,K,zi,b,c)
% function [r,zf]=rice1(N,K,zi,b,c)
% returns N Rician unit variance variables. Z0 is the initial sample for the IIR
filter
% a is the ratio of the fading rate to the sampling frequency in Hz
% based on lmfading.m of R Young
      % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

if sum(abs(zi))==0
   zi=[zeros(max(length(b),length(c))-1,1)];
 end

r=(1/sqrt(2))*(randn(N,1)+j*randn(N,1));
[r,zf]=filter(b,c,r,zi);
sigma=10^(-K/20);
r=1+sigma*r;
```

addseq.m

```
% addseq
% W F McGee
% contract U6800-9-0526
% 31 Jan 1999
                  databl=datgen(dlen,numchan);  %Compute a blen x numchan matrix
of random
                                                                    %+1 and -1.
            datablsave=databl;

databl=[databl(1:floor(dlen/2),:);zeros(slen,numchan);databl((floor(dlen/2)+1):d
len,:)]; %stick in the
                % blanks

         lintree % Calculates the signal sequence at the output; returns sig
         sigtot=sig ;

         spillover=sig(start:(start+31));
         correction=A\spillover;
         extra=extra+correction'*correction;
         esave=esave+spillover.*spillover;
         nsave=nsave+1;
         % second time through the bank with the inserted symbols
         databl=[];
         for i=1:numchan
            databl=[databl correction((1+(i-1)*slen):(slen+(i-1)*slen))];
         end
         lintree;

         sigdiff=sigtot;
         for i=1:length(sig)
            sigdiff(i+numchan*floor(dlen/2))=sigdiff(i+numchan*floor(dlen/2))-
sig(i); % subtract the two
         end
         sigdiff;
         sigtot=sigdiff;
         for i=1:length(mseq)
            sigtot(start+i-1)=sigtot(start+i-1)+mseq(i); % insert the m sequence
         end
```

makesqam.m

```
function [b]=makesqam(b,c);
% function [b]=makesqam(b,c);
% clobbers b
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

x=1;
for i=1:length(b)
   b(i)=b(i)*x;
   x=x*c;
end
```

chancgn.n

```
function y=chancgn(sig1,esno)
%
% chanccgn adds circularly complex white gaussian noise to a
% signal vector.  It is used after the bandpass version of the synthesis filter
bank
% at the transmitter end.
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999
% from Mike Sablatash program
sigma=1./sqrt((10.^(esno/10.))*2);  % compute the noise deviation
sp1=size(sig1);
j=sqrt(-1);
y=sig1+sigma*(randn(sp1(1),sp1(2))+j*randn(sp1(1),sp1(2)));
```

getphase4.m

```
function
[chanest1,chanest2,chanest3,t1,t2,t3]=getphase4(sigtotal,mseq,offset,spacing)
% getphase
% matched filter, and picks the peak
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

v=conv(flipud(mseq),sigtotal);
chanest1=v(offset)/32;
chanest2=v(offset+spacing)/32;
chanest3=v(offset+2*spacing)/32;
t1=gett(v,offset);
t2=gett(v,offset+spacing);
t3=gett(v,offset+2*spacing);
return

function t=gett(v,offset);
t=abs(v(offset+1))-abs(v(offset-1));
d=2*abs(v(offset))-abs(v(offset+1))-abs(v(offset-1));
t=0.5*t/d;
return
```

myinterp4.m

```
function [a]=myinterp4(y1,y2,y3,offset,spacing,len)
% function [a]=myinterp4(y1,y2,y3,offset,spacing,len)
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

k=1:len;
f1=y1/(2*spacing*spacing);
f2=-y2/(spacing*spacing);
f3=y3/(2*spacing*spacing);
x1=offset;
x2=x1+spacing;
x3=x2+spacing;
a=f1*(k-x2).*(k-x3)+f2*(k-x1).*(k-x3)+f3*(k-x1).*(k-x2);
```

untree.m

```
% detree forms data from sigtotal
% clobbers sigtotal
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

sigtotal=[0;sigtotal]; % take the odd samples
if is3a
   data1=[upfirdn(sigtotal,f3,1,2) -upfirdn(sigtotal,hf3,1,2)];
else
   data1=[upfirdn(sigtotal,ht3,1,2) upfirdn(sigtotal,t3,1,2)];
end
data=data1;
if is2a
   data1=[upfirdn(data(:,1),f2,1,2) -upfirdn(data(:,1),hf2,1,2)];
else
   data1=[upfirdn(data(:,1),ht2,1,2) upfirdn(data(:,1),t2,1,2)];
end
if is2b
   data2=[-upfirdn(data(:,2),hf2,1,2) upfirdn(data(:,2),f2,1,2)];
else
   data2=[upfirdn(data(:,2),t2,1,2) upfirdn(data(:,2),ht2,1,2)];
end
data=[data1 data2];
if is1a
   data1=[upfirdn(data(:,1),f1,1,2) -upfirdn(data(:,1),hf1,1,2)];
else
   data1=[upfirdn(data(:,1),ht1,1,2) upfirdn(data(:,1),t1,1,2)];
end
if is1b
   data2=[-upfirdn(data(:,2),hf1,1,2) upfirdn(data(:,2),f1,1,2)];
else
   data2=[upfirdn(data(:,2),t1,1,2) upfirdn(data(:,2),ht1,1,2)];
end
if is1c
   data3=[upfirdn(data(:,3),f1,1,2) -upfirdn(data(:,3),hf1,1,2)];
else
   data3=[upfirdn(data(:,3),ht1,1,2) upfirdn(data(:,3),t1,1,2)];
end
if is1d
   data4=[-upfirdn(data(:,4),hf1,1,2) upfirdn(data(:,4),f1,1,2)];
else
   data4=[upfirdn(data(:,4),t1,1,2) upfirdn(data(:,4),ht1,1,2)];
end
data=[data1 data2 data3 data4];
datasize=size(data);
data=data(2:datasize(1),:);
```

decode.m

```
function [a]=decode(b)
a=zeros(size(b));
bsize=size(b);
for i=1:bsize(1)
   for k=1:bsize(2)
      if b(i,k)>0
         a(i,k)=1;
      else
         a(i,k)=-1;
      end
   end
end
```

ricetheory.m

```
function [result]=ricetheory(SNR,RiceFactor);
% determines the error rate for BPSK with Rice fading K
% calculates error rate for Rice with perfect phase correction
% integration over 100 points
 % W F McGee
% contract U6800-9-0526
% 31 Jan 1999

range=pi/2;
num=100;
df=range/num;
f=df/2;
rice=10^(-RiceFactor/10);
sigma=10^(-SNR/10);
sum=0;
for i=1:(num)
   a=sin(f);
   a=a*a;
   f1=rice+sigma*a;
   f2=sigma*a;
   dsum=df*exp(-1.0/f1)*f2/f1;
   sum=sum+dsum;
   f=f+df;
end
sum=sum/pi;
result=sum;
```

List of Figures

List of Tables

List of MATLAB files (all .m files unless noted)

```
contents: list of input variables, output, required files
rsim3:  main program
q: normal probability
plotpe: plot error rate vs SNR
maketree: generates filters needed for simulation tree
high: makes high pass filter from lowpass
f1.mat: filter coefficients (c) W F McGee 1998
f2.mat:      "
f3.mat:      "
valid.mat: definition of valid tree configurations
makeAmatrix: generates the A matrix used in addseq
lmffilt1: generates an IIR filter coefficients for channel simulation
NoiseBW: used to calibrate IIR filter
datgen: generates random data
nrequired: the number of error events required for pretty curves
rice3: generates a vector of Rician channel values
addseq: generates random data, filters it, measures residue,
            inserts data to cancel, inserts mseq
lintree: does the tree signal processing
makesqam: transforms a sequence into an offset QPSK sequence
chancgn: Gaussian noise generator
getphase4: determines channel estimates from matched filter
myinterp4: interpolates a channel estimate
untree: analysis tree
decode: recovers the input data estimate
ricetheory: theoretical error rate for a Rician channel, perfect sync
```