DESIGN AND IMPLEMENTATION OF A 4-LEVEL SOFT-DECISION VITERBI DECODER AT A DATA RATE OF 2.048 Mbit/s.

by W.H.C. de Krom.


Professor : prof. Ir. J. van der Plaats
Supervisor : Ir. A.P. Verlijjsdonk

The faculty of electrical engineering of the Eindhoven University of Technology disclaims any responsibility for the contents of training and graduation reports.
Contents.

Summary ........................................................................................................ 1

List of symbols ................................................................. 2

1. Introduction ........................................................................ 3

2. Review of convolutional coding and hard/soft-decision Viterbi decoding ......................... 5

   2.1 Convolutional coding .................................................. 5
   2.2 The Viterbi decoding algorithm ................................. 8

3. An efficient way to transmit a pseudo-ternary signal ................................................. 17

   3.1.1 3ASK modulation ............................................. 17
   3.1.2 Trellis coded modulation ................................... 17

   3.2 Comparisons of several codulation systems .................. 21

      3.2.1 Codulation systems with QPSK and 8PSK modulation versus uncoded BPSK modulation .......... 21
      3.2.2 Codulation system with 8PSK modulation versus uncoded 3PSK modulation ......................... 28
      3.2.3 The complexity ................................................. 32

4. The implementation of a 4-level soft-decision Viterbi decoder, operating at an encode bit rate of 4.096 Mbit/s ................................................. 34

   4.1 Model of the communication system ......................... 34
4.2 The general structure of the decoder, including the A/D converter

4.2.1 The encoder

4.2.2 The Viterbi decoder

5. Testing procedure and conclusions

5.1 Testing procedure

5.1.2 Results and conclusions

6. Conclusions

Acknowledgement

References

Appendix A Block diagram of the Viterbi decoder
Appendix B1 Input section
Appendix B2 Branch metric section
Appendix B3 ACS section
Appendix B4 Path memory
Appendix B5 Output section
Appendix B6 Clock control section
Appendix C1 Pascal program BER
Appendix C2 Pascal program BMC + ACS
SUMMARY.

In some rural communication systems a pseudo-ternary converted primary multiplex PCM signal with a data rate of 2.048 Mbit/s, has to be transmitted. Often this is accomplished by means of BPSK modulation, which offers a good balance between bandwidth utilization and system complexity. For rural communication systems costs are more important than bandwidth utilization. For this reason BPSK modulation has been an appropriate choice. But nowadays, even in rural communications the need grows for communication systems, which are more power and bandwidth efficient, but still of moderate complexity.

This report describes the design of such a more efficient communication system as well as the development and realization of the coder/decoder part of a modem, operating at an uncoded data rate of 2.048 Mbit/s.

A rate $R = 1/2$ and constraint length $K = 3$ trellis coded QPSK modulation with soft-decision detection, results in a powerful, so called, codulation system. This codulation system achieves a theoretical coding gain versus uncoded BPSK modulation of 4 dB. This gain is derived under the assumption of unlimited coding and decoding effort.

Since the implemented coding procedure doubles the input data rate to a channel rate of 4.096 Mbit/s, the main problem was the realization of a 4-level soft-decision Viterbi decoder, able to work reliable at this rate.

The performance of the realized Viterbi decoder in terms of bit error rate is compared with the theoretical results, based on the characteristics of the realized decoder. A computer program calculates the upper bound of the bit-error rate for the 4-level soft-decision Viterbi decoder as a function of the ratio $E_b/N_0$.

The measured performance of the Viterbi decoder matched the analytical derived results very well. It turned out, that the measured coding gain of approx. 3.3 dB for $P_b \leq 10^{-5}$, hardly differs from the maximum possible theoretical coding gain.

The maximum possible measured encoded data rate on which the decoder still operates reliable, appears to be about 10 Mbit/s.
List of symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACS</td>
<td>add compare select</td>
</tr>
<tr>
<td>ASK</td>
<td>amplitude shift keying</td>
</tr>
<tr>
<td>AWGN</td>
<td>additive white Gaussian noise</td>
</tr>
<tr>
<td>b</td>
<td>number of bits shifted in the encoder at a time</td>
</tr>
<tr>
<td>BM</td>
<td>branch metric</td>
</tr>
<tr>
<td>BMC</td>
<td>branch metric calculation</td>
</tr>
<tr>
<td>BER</td>
<td>bit error rate</td>
</tr>
<tr>
<td>BPSK</td>
<td>binary phase shift keying</td>
</tr>
<tr>
<td>$d_{\text{free}}$</td>
<td>minimum free Euclidean distance</td>
</tr>
<tr>
<td>DMC</td>
<td>discrete memoryless channel</td>
</tr>
<tr>
<td>ED</td>
<td>Euclidean distance</td>
</tr>
<tr>
<td>$E_s$</td>
<td>signal power of encoded symbol</td>
</tr>
<tr>
<td>HD</td>
<td>Hamming distance</td>
</tr>
<tr>
<td>HDB-3</td>
<td>high density binary encoded symbols</td>
</tr>
<tr>
<td>n</td>
<td>sampled noise value</td>
</tr>
<tr>
<td>$N_o$</td>
<td>noise power density (double sided)</td>
</tr>
<tr>
<td>OS</td>
<td>output section</td>
</tr>
<tr>
<td>P</td>
<td>survivor path</td>
</tr>
<tr>
<td>$P_b$</td>
<td>bit error probability (of Viterbi decoder)</td>
</tr>
<tr>
<td>PM</td>
<td>path memory</td>
</tr>
<tr>
<td>QPSK</td>
<td>quadrature phase shift keying</td>
</tr>
<tr>
<td>r</td>
<td>sampled value of output demodulator</td>
</tr>
<tr>
<td>R</td>
<td>rate of convolutional encoder</td>
</tr>
<tr>
<td>S</td>
<td>signal power</td>
</tr>
<tr>
<td>$s_i$</td>
<td>state 1 in trellis diagram</td>
</tr>
<tr>
<td>SM</td>
<td>state metric</td>
</tr>
<tr>
<td>SMC</td>
<td>state metric calculation</td>
</tr>
<tr>
<td>SNR</td>
<td>signal to noise ration</td>
</tr>
<tr>
<td>sp</td>
<td>signal point</td>
</tr>
<tr>
<td>T</td>
<td>normalized threshold value</td>
</tr>
<tr>
<td>$z_i$</td>
<td>output value of encoder</td>
</tr>
<tr>
<td>3PSK</td>
<td>three phase shift keying</td>
</tr>
<tr>
<td>4B3T</td>
<td>four binary to three ternary encoded symbols</td>
</tr>
<tr>
<td>8PSK</td>
<td>eight phase shift keying</td>
</tr>
<tr>
<td>9PSK</td>
<td>nine phase shift keying</td>
</tr>
</tbody>
</table>
1. Introduction.

In rural communications it sometimes happens that a primary multiplex PCM signal of 2.048 Mbit/s, coded as a pseudo-ternary signal, has to be transmitted by means of a radiolink. There are several ways of doing this, all with their own advantages and disadvantages.

For example:

1. converting the pseudo-ternary signal into an uncoded binary signal and transmission with BPSK (binary phase-shift keying) or QPSK (quadrature phase-shift keying).

2. direct transmission with three-level ASK (amplitude shift-keying) or with three-phase PSK (phase shift-keying).

3. using e.g. a convolutional encoder for channel coding. This creates the opportunity to transmit encoded signals with a certain amount of redundancy. By doing so, we are able to reduce the bit error rate. At the receiver side we use coherent demodulation followed by a Viterbi decoder.

An important figure of a digital transmission system is the number of transmitted bits per second per Hertz bandwidth (b/s/Hz). However, this so called bandwidth efficiency is not the only criterion for a good digital communication system. The ultimate goal is to achieve a low bit error probability (P(e)), with an energy per bit to noise-density ratio (Eb/No) requirement as low as possible in an interference environment (noise and phase jitter). Other factors which also must be taken into account in order to design an optimal communication system are:

- hardware complexity,
- power efficiency,
- costs and
- availability.
The modulation method most frequently used in the past two decades for the transmission of a pseudo-ternary primary multiplex PCM signal in rural communications have been BPSK, preceded by a 3-to-2 level converter. This, because BPSK offers a balance between bandwidth utilization and system complexity. The fact, that there were no strict bandwidth restrictions, justifies this choice for using a system of minor complexity.

However as the need for more efficient communication systems continues, we have to pay more attention to other modulation methods, with higher bandwidth efficiency than BPSK.

The first part of this report describes an efficient modulation scheme to transmit a pseudo-ternary signal over a radiolink in rural areas. The second part describes an implementation of the coder/decoder part of the modem for this particular modulation scheme i.e., a 4-level soft-decision Viterbi decoder, with a bit rate of 2.048 Mbit/s.
CHAPTER 2.


2.1 Convolutional coding.

Historically, the coding systems have been divided into two techniques, i.e. block coding and convolutional coding. In an (n,k) linear block code a sequence of k bits is extended with n-k redundant code bits to give an overall encoded block code of n bits.

Linear codes have the very important property, that from a set of code words two arbitrary code words can be added to produce a third code word belonging to the same set. An other way of implementing this property, is the fact that there is no loss in generality in computing the (Hamming or Euclidean) distance from the all-zero code word to all the other code words, since this set of distances is the same as the set of distances from any arbitrary code word to all the others.

The code rate of a block code is \( R = k/n \), where n is called the block length. Note that the introduction of redundant bits in order to provide error-correcting properties, requires more transmission capacity.

Convolutional codes are generated by a convolutional encoder (figure 2.1).

![Figure 2.1: A convolutional encoder with K=3, b=1 and n=2.](image-url)

Figure 2.1 A convolutional encoder with K=3, b=1 and n=2.
In general a linear finite-state machine consists of a $K$-stage shift register and $n$ modulo-2 adders. The latter simply perform the exclusive-or operations in digital logic. Each of the modulo-2 adders is connected to particular memory elements of the register. The pattern of these connections specifies the code. The input data are shifted into the register with $b$ bits at a time ($b < n$). The number of shift register stages multiplied by $b$ is called the constraint length $K$, since that is the number of output bits which are dependent from one single information bit. Convolutional codes are a subset of the linear codes and perform better than block codes of the same complexity.

The possible sequences of output bits are conveniently displayed by means of a tree diagram as shown in figure 2.2. Each branch of the tree represents the output related to a particular set of 3 bits (including the real time information bit(s)) in the shift register.
Now notice that starting e.g. with the 3th branch of any possible sequence, there are identical sets of branches in the upper and lower halves of the tree diagram. This fact implies that it must be possible to tie nodes together in order to get a more compact tree without losing any new information. Thus at each depth within the tree beyond the depth corresponding to the constraint length of the code, the number of branches can be halved. By following this recipe, we will get a repetitive structure called a trellis (figure 2.3).
This trellis will appear to be very useful for decoding the data sequence by means of the Viterbi algorithm, which will be explained in the next section.

2.2 The Viterbi algorithm.

The Viterbi (or maximum-likelihood) decoding algorithm is a powerful technique for decoding convolutionally encoded data. It was discovered and analyzed by Viterbi [1] in 1967. As with block codes we are interested in finding that transmitted sequence, which has the greatest probability of being transmitted by the transmitter.

A conventional maximum-likelihood decoder would choose that sequence whose encoded version is in some way closest to the received sequence. Usually the Hamming or Euclidean distance is being taken as a measure of closeness or similarity of two sequences. Note that the number of paths is dependent of the message length. Assume we have a message length of $B$ bits. This results in $2^B$ paths, so decoding might seem to become almost impossible for large $B$.

The Viterbi algorithm.

The Viterbi algorithm is able to control the maximum number of paths, because it makes use of the special periodic structure of the trellis diagram. By doing this, the complexity of the decoder only depends on the constraint length $K$ of the encoder rather than on the message length.
Unfortunately the larger the $K$ is the better the code is likely to be, that is the larger the coding gain that can be obtained.

For example consider again the trellis diagram for the encoder of figure 2.1. in figure 2.3.

![Trellis Diagram Image]

*Figure 2.4 The code trellis diagram.*

For state 00 at depth 3 in the trellis diagram, two paths are entering corresponding to the information sequences 000 and 100 (corresponding to the upper resp. lower thick printed path in the trellis diagram). One of these paths has diverged in a previous state from the all-zero sequence and merged after a several transitions. The task of the Viterbi algorithm is to calculate the likelihood (is metric) of each of those incoming paths and select the most likely one, called the survivor. The other path is eliminated. This procedure is followed for every state at a given trellis depth. In this way the algorithm is able to control the number of paths that have to be remembered, because after each decoding operation only one path (the most likely one), leading to each state at a certain depth in the trellis, remains. When the decoding algorithm has calculated the $2^{K-1}$ metrics, functions which express the likelihood of a certain state, it will repeat the procedure one level deeper in the trellis.

Note, that when two sequences merge in a certain state having the same metric, the decoder will not be able what to decide. Fortunately it appears to be unimportant for the average error probability, for which one is decided, because further received sequences (symbols) will affect both metrics in the same way. So the decoding algorithm should for example take in this case the predefined upper or lower incoming branch.
Analytical description of the Viterbi algorithm.

As explained in the latter section, the decoding consists primarily of calculating within each received symbol time a new metric for each of (in our case) four states, called the state metric. The weight of each state metric is a measure of likelihood, that a particular data sequence has been transmitted, ending in that particular state.

This procedure operates as follows. The branch metric for each of (in our case) two possible branches combining in a given state is computed and added to the state metric corresponding to the state from which these branches originated.

The addition of each branch metric to its previous state metric, results in two path metrics per state. The largest of the two path metrics is selected as the new state metric if we use the Viterbi metric \[1\) (the negative log-likelihood function). The smallest of the two is selected if we defined the metric to be a distance metric (path with the smallest distance metric has the largest likelihood of being sent).

We now shall write this in analytical form in order to be able to implement a decoder in hardware.

Let \( t \in T = \{1,2,3,\ldots\} \) represent time and let \( s_1,s_2 \in S \) represent states of the trellis diagram in figure 2.3. The path metric of a state \( s_1 \) at time \( t \) is called \( \text{PM}_{s_1}(t) \). The branch metric, defines as the distance metric of a received data word to the four possible transitions from a state \( s_1 \) at time \( (t-1) \) to a state \( s_2 \) at time \( t \), is called \( \text{BM}_{s_1s_2}(t) \). The state metric of a state \( s_1 \) at time \( t \) is called \( \text{SM}_{s_1}(t) \).

The main function the Viterbi algorithm has to perform is:

\[
\text{SM}_{s_1}(t) = \min \{ \text{PM}_{s_1s_2}(t), \text{SM}_{s_1}(t-1) + \text{BM}_{s_1s_2}(t) \} \quad s_1,s_2 \in S \text{ and } t \in T.
\]

where \( \text{PM}_{s_1s_2}(t) \) are the (in our case) two possible path metrics for state \( s_2 \) at time \( t \). In analytical form we can write:

\[
\text{PM}_{s_1s_2}(t) = \{ \text{SM}_{s_1}(t-1) + \text{BM}_{s_1s_2}(t) \} \quad s_1,s_2 \in S \text{ and } t \in T.
\]
Along with the calculation of the path metrics is SMs(t+1) for each state, the algorithm also has to store the survivor path for each state, called Ps(t). In analytical form:

\[
Ps(t) = \sum_{t'=0}^{t} Ms(t') \quad s_2 \in S
\]

The output of the decoder is that path, which has the minimum overall (distance) metric at a certain time \( t \in T \).

**Hard- and soft-decision with Viterbi decoding.**

Until now we have told nothing about the need for quantization. In practical communication systems we are almost never able to process the actual sampled input voltage of the decoder, so quantizing is an important step prior to decoding. In this case we have the so called binary symmetric channel (BSC) with an error probability of \( P_e \) shown in figure 2.5.

![Figure 2.5](image)

**figure 2.5** The binary symmetric channel (hard-decision quantizing).

If a binary (two level) quantization is used we call this hard decision. When coding is used, hard quantization of the received data usually results in a loss of about 2 db in \( E_b/N_0 \) compared with infinitely fine quantization. Much of this loss can be avoided by quantizing the received
data into 4 or 8 levels instead of only two. The channel resulting from two- or three-bit quantization on a Gaussian channel is called the binary input, 8-ary output discrete memoryless channel (DMC), and is shown in figure 2.6.

A demodulator operating in this way, is called a soft-decision demodulator. It decides whether the sampled value of the received data signal is above or under the quantizing threshold. Next it computes a three bit code (for 8-level quantization), which specifies in a special code, how much the sample value differs from the zero level. In this way the Viterbi decoder will get more information about the likelihood of a certain data bit being transmitted and we are able to minimize the loss of 2 dB for hard quantization to about 0.25 dB for 8-level quantization and to about 0.7 dB for 4-level quantization. The hardware however will in the latter case be less complex and easier to implement at high speed, than for 8-level quantization.

As will be shown in a next section, the Viterbi decoding algorithm can easily operate on soft-decision channel symbols without a major increase in complexity.

figure 2.6 A 2-input, 8-ary output DMC.
The BER performance of a soft-decision Viterbi decoder, with 4-level quantization.

In this section we address the problem of finding the bit-error-rate performance of the Viterbi decoder. To gain more insight in this aspect, the reader is advised to read two references. The first one is by Viterbi [1] and discusses the basic properties of convolutional codes and their performance. The second paper is by Yasuda, Hirata and Muratani [2] and it discusses the basic aspects of calculating the bit error probability for a soft-decision Viterbi decoder. To develop an algorithm for the computation of the bit error probability, we will quote some results from these two references.

The transmission channel model considered here is shown in figure 2.7.

*figure 2.7 The transmission channel model.*
The sampled value $r$ of the baseband signal at the output of the demodulator can be represented by:

$$r = \pm \sqrt{E_s} + n$$  \hspace{1cm} (1)

where $E_s$ is the signal energy per coded symbol and $n$ is the white Gaussian noise with variance $N_o/2$. As discussed in the section above, the transmission channel for a 4-level soft-decision Viterbi decoder, can be regarded as a binary input, 4-ary output symmetric memoryless channel. The channel's transmission probability $P(1)$ is defined as the probability, that the output symbol $y_i$ is received when the input symbol $x = 1$ is transmitted. Therefore $P(1)$ can easily be calculated as a function of $E_s/N_o$ and the equally spaced thresholds $T_i \{i = 1,2,3,4\}$:

$$P(1) = Pr(y = y_1 \mid x = 1) = Pr (y = y_{Q+1-1} \mid x = 0)$$

$$= \frac{1}{\sqrt{N_o} \pi} \int_{T_i}^{1+1} \exp \left[ -\frac{(r - \sqrt{E_s})^2}{N_o} \right] dr \hspace{1cm} 1 = \{1,2,3,4\}$$ \hspace{1cm} (2)

$T_i$ is related to $\sqrt{E_s}$

The Viterbi upper bound for the bit error probability, when used a convolutional encoder with rate $k/n$, is tightly upper bounded by [1]:

$$P_b \leq \frac{1}{k} \sum_{k=d}^{\infty} C_k P_k$$  \hspace{1cm} (3)

where $d$ is the free distance of the used convolutional code. $C_k$ is the total number of erroneous bits included in all possible incorrect paths, whose distance from the correct path equals $k$. $P_k$ is the probability that one of such incorrect paths is selected as the most likely one during the decoding process.

The calculation of $C_k$:

If we assume the correct path to be the all-zero sequence, then $C_k$ is the number of paths with distance $k$ multiplied by the number of "1's" in the received data sequence. With this in mind, $C_k$ can be calculated by using the generating function $T(D,L,N)$, introduced by Viterbi [1], by which all the properties of the used convolutional code are defined.
For example the generating function for the coder of figure 2.1 is:

\[ T(D,N) = T(D,L,N) \mid L=1 = \frac{D^N}{1 - 2DN} \]

where

- the power of \( D \) is the distance of a path,
- the power of \( L \) is the length of a path,
- The power of \( N \) is the number of "1" input bits.

Because we are only interested in the number of erroneous paths and not their length, \( L \) is chosen to be 1.

\[ T(D, N) = \sum_{d=5}^{\infty} 2^{d-5} D^d N^{d-4} \]

(4)

\( C_k \) can now be expressed by means of the generating function as follows:

\[ \frac{dT(D,N)}{dN} = \sum_{k=d}^{\infty} (d-4).2^{d-5} D^d N^{d-5} = \sum_{k=d}^{\infty} C_k D^d N^{d-5} \]

(5)

For the calculation of \( P_k \), the following relations can be derived [2]:

\[ P_k = \sum_{n=-\infty}^{-1} q_k(n) + 0.5q_k(0) \]

(6)

where

\[ q_k(n) = \sum \frac{k!}{11! \cdot 12! \cdot 13! \cdot 14!} \cdot P(1)^{11} \cdot P(2)^{12} \cdot P(3)^{13} \cdot P(4)^{14} \]

(7)

The summation \( \sum \) means the summation for all set of integers \( \{11\} \), which satisfy the following conditions:

\[ \sum_{i=1}^{Q} (1 \times 11) = \frac{(Q+1) \cdot k - n}{2} \]

(8)

\( Q = 4 \), the number of quantization levels.
The bit error probability $P_b$ is bounded by several infinite weighted summations of $P_k$. Because the fact, that computation of an infinite summation is impossible, we have to truncate the summations at, for our realization of the Viterbi decoder, sufficiently large values for $k$ and $n$. This will be discussed in chapter 4.

Unknown starting state of the decoder.

In the preceding it has been assumed that a Viterbi decoder has knowledge of the encoder starting state before coding begins. A known starting state is for the hardware implementation of the Viterbi decoder not very attractive, because this requires that the decoder knows when a data transmission starts. An other way of solving this problem is to introduce a prefix code. This can be the all-zero code word. If the transmitter start with this prefix code before the actual information is transmitted, the decoder is forced to go to the 00 state.

In reality however, this problem is less complex. It has been found through computer simulations that a Viterbi decoder may start processing the received data at any arbitrary state under only one condition, i.e. all the state metrics have to be reset to zero before the decoding algorithm starts. The result of the unknown starting state is, that the first 3 to 4 constraint length data words will be more or less unreliable [3]. However after 4 constraint lengths the state metrics will have values, which are independent of the starting state and the result is a highly reliable decoded data sequence.
CHAPTER 3.

An efficient way to transmit a pseudo-ternary signal.

3.1.1 3ASK modulation.

The in the introduction mentioned possibility of transmitting a pseudo-ternary signal by means of 3ASK modulation is not very appropriate. Firstly, this modulation method is power inefficient. Secondly, its performance degrades by non-linearities in system components and noise. Thirdly the complexity is still rather involved. We therefore put our main attention towards systems using phase shift keying modulation techniques.

3.1.2 Trellis coded modulation.

Trellis coded modulation (TCM) has evolved over the past few decades as a combined coding and modulation technique for digital transmission over a bandwidth limited channel. Because of the fact that, in the contrary to the past, modulation and coding are considered as a whole, it is often called codulation (COding + moDULATION) [4] [5] [6].

The following properties make TCM very powerful:

1. it allows significant coding gains over conventional uncoded multi-level modulation, having the same BER performance,

2. no bandwidth expansion is necessary to introduce redundant coding bits,

3. the effective information rate is not affected during the coding process.

The TCM system consists of a convolutional encoder, a finite-state machine, which defines the selection of the channel symbols in relation to the current and past input data.
In the receiver the unquantized demodulator output signal, added with white Gaussian noise, is decoded by means of a soft-decision maximum-likelihood decoder, which usually applies the Viterbi algorithm. The fact that the demodulator output is not quantized, implies that codes for multi-phase signals should be designed to achieve maximum free Euclidean distance rather than Hamming distance.

The new basic concept of TCM, introduced by Ungerboeck in 1982 [4], is the use of signal set expansion to provide redundancy for coding without an increase of bandwidth, followed by a signal points mapping function to maximize the free (Euclidean) distance between all possible coded signal sequences. This results in constructions of codulation schemes, whose free (Euclidean) distance significantly exceeds the minimum free distance between two arbitrary uncoded modulation sequences. Since a larger Euclidean distance implies a smaller bit error probability at the same signal-to-noise ratio, TCM introduces a gain in signal power.

In order to get a better understanding of the basic concept of TCM, we first consider figure 3.1.

![Channel capacity of bandlimited AWGN channels](image)

**figure 3.1** The channel capacity of bandlimited AWGN channels with discrete-valued input and continuous-valued output. [4]
From figure 3.1 can be concluded, that the transmission of 1 bit/T by uncoded BPSK (2-PSK) modulation with $P_e = 10^{-6}$, occurs at a SNR of approximately 9.5 dB. If the number of possible signal points is doubled e.g. by choosing QPSK modulation, almost error-free transmission of 1 bit/T is theoretically possible already at a SNR of approx. 0.5 dB (assuming unlimited coding and decoding effort). Beyond this, with no constraint on the number of signal points (is number of phases) except average signal power, only 0.3 to 0.4 dB can further be gained. Now it can be concluded, that by doubling the number of signal points almost all possible is gained in terms of channel capacity for a small SNR.

From the preceding we can conclude, that in order to transmit $m$ bits/T in redundantly coded form, we must have a set of $2^m + 1$ signal points (equal to number of phases). This can be accomplished by a convolutional encoder of rate $R = m/(m+1)$, followed by a mapping function, which maps the $m+1$ bits into the larger set of signal points in the most optimum way to guarantee a maximum possible Euclidean distance (see figure 3.2). Notice, that the number of possible signal points is now larger than in the uncoded case.

In summary the task of the convolutional encoder is to select signal points in such a way that:

1. the number of possible transitions in a trellis diagram is limited, while

2. the minimum free Euclidean distance $d_{\text{free}}$ between all pairs of signal points sequences $\{a_n\}$ and $\{a'_n\}$, which the encoder can produce is maximum.

Where

$$d_{\text{free}} = \min \left[ \sum_{n} d^2(a_n, a'_n) \right]^{1/2} \quad \text{for each } a_n \neq a'_n \quad (1)$$

and

$$d^2(a_n, a'_n) = \sum_{i} \| a_{n_i} - a'_{n_i} \|^2 \quad (2)$$

19
If soft-decision maximum likelihood decoding is applied, the bit error probability $P_b$ will approach asymptotically at high SNR to the lower bound:

$$P_b \geq N(d_o) \cdot Q(d_o/(2\sigma)) \tag{3}$$

where

$$d_o = \text{the minimum Euclidean distance between any two different signal points,}$$

$$N(d_o) = \text{the average number of neighbouring signal points at the minimum Euclidean distance } d_o \text{ of a given signal point.}$$

It is obvious that a large value of $d_o$ is desirable to obtain a low error probability $P_b$. Simply increasing the Euclidean distance between the signal points is not the best technique. This because of the fact that the average energy of the two-dimensional signal points $E_s$ is related to the Euclidean distance.

$$E_{sp}(x_1, x_2) = (x_1^2 + x_2^2). \tag{4}$$

The procedure to find the optimum mapping function, which is the same as assigning the signal points to the transitions in the trellis diagram is called "mapping by set partitioning". This mapping follows from successive partitioning of the set of signal points into smaller subsets with increasing distance between the signal points of these subsets under the condition, that $E(a_n^2) = 1$ (is normalized). The doubling of the number of channel symbols results for an uncoded PSK system in a higher error probability. This because the minimum Euclidean distance between two signal points becomes smaller. The by means of set partitioning created subsets of signal points will be assigned to the states of the encoder. This in combination with the fact that the convolutional encoder limits the number of originating transitions from a certain state, will result in a greater free distance between two arbitrary signal sequences.
According to this strategy the signal points have to be assigned to the transitions in the trellis in the following way:

1. Whenever two or more transitions diverge or merge in a certain state, we have to assign signal points to these transitions, which belong to the same subset (say A). In this case, all signal sequences will have a squared Euclidean distance of at least twice the squared minimum intra-set distance of A. This can be explained as follows. The shortest possible excursion from the correct sequence consists of at least two transitions. For each transition the squared Euclidean distance from the correct path is equal to the minimum intra-subset distance. Therefore the total excursion has a distance from the correct path of at least twice the squared intra-subset distance.

2. If possible, parallel transitions should be avoided, since they limit the free distance and for this reason the coding gain. If they cannot be avoided we must assign signal points from the same subset to these parallel transitions. This results in maximum possible distance between both transitions.

3.2 Comparisons of several codulation systems.

3.2.1 Codulation systems with QPSK and BPSK modulation versus uncoded BPSK modulation.

In this section we will investigate several codulation systems and compare them with the BPSK modulation system preceded, by a 3-to-2 level converter. The BPSK system is taken as reference, because this is the system already applied for transmission of a pseudo-ternary signal in some rural communication systems. On the other hand it offers a reasonable and fundamental balance between bandwidth utilization and system complexity (figure 3.2).
The comparisions are strictly made on the basis of equal bandwidth and data rate. For the following codulation systems the gain versus the BPSK system is calculated, using an optimal convolutional encoder with a minimum number of states, in order to limit the complexity of the Viterbi decoder. There will also be given an indication about the expected complexity of the modulation systems for trade-offs in a following section.

Considered are:

- Coded QPSK with convolutional encoder of rate $R = 1/2$,
- Coded 8PSK with convolutional encoder of rate $R = 1/2$,
- The codulation system with QPSK modulation.

The codulation system considered in this section, is given in figure 3.3.

The convolutional encoder is a four state machine, because the requirements for signal point assignment, given in a previous section, can not be fulfilled by means of a two state encoder. This can easily be verified. The used code is a systematic code with rate $R = 1/2$ and maximum free distance. All possible transitions from state $s_{12}$ at $t=t_1$
to state $s_1 s_2$ at time $t_1 < t_2$ of the convolutional encoder are given in the trellis diagram in figure 3.4, together with the corresponding values of the outputs $z_0 z_1$ during the transition. Since for a given state $s_1 s_2$, a transition takes place while output $z_0$ equals $s_1$, the number of possible output values during a transition from a given state is limited.

![Figure 3.4-1 The used convolutional encoder.](image)

The signal points now have to be assigned to the transition in such a way as to assure maximum free distance. By set partitioning of the QPSK signal points we are able to do so (figure 3.5).

![Figure 3.4-2 The trellis diagram.](image)
figure 3.5 Partitioning of the QPSK signal points.

From figure 3.5 we see that by letting \( z_0 \) select the subset and \( z_1 \) the signal points in this subset we achieve the optimal assignment of the signal points to the transitions.

Suppose the all-zero sequence is transmitted. The shortest, from this sequence diverging at time \( t_1 \) and merging at time \( t_2 \) path, is the path printed thick in the trellis diagram of figure 3.4.

The distance from the all-zero sequence now equals:

\[
d_{\text{free}} = \sqrt{d^2(\text{sp } 0, \text{sp } 2) + d^2(\text{sp } 0, \text{sp } 1) + d^2(\text{sp } 0, \text{sp } 2)}
\]

\[
= \sqrt{(2^2 + (1/2\sqrt{2})^2 + 2^2)} = \sqrt{10} = 3.16
\]

where \( d^2(\text{sp } p, \text{sp } q) \) is the squared Euclidean distance between the signal points \( p \) and \( q \) under the condition that \( E(p^2) = E(q^2) = 1 \).

The gain versus uncoded BPSK modulation can now be calculated according
to the following relation:

\[
\text{gain} = 10 \cdot 10^{\log \left( \frac{d_{\text{free}}^2}{d_0^2} \right)} + 10 \cdot 10^{\log \left( \frac{E_{\text{sp}}}{E_{\text{sp coded}}} \right)} \quad (5)
\]

where

- \(d_0\) = the maximum distance of the uncoded signal points of the BPSK modulation system under the same condition as \(d_{\text{free}} = 2\).
- \(E_{\text{sp}}\) = is the energy of a signal point of the BPSK signal constellation, which is not affected by the coding process. This because \(E_{\text{sp}}\) for QPSK and BPSK modulation are equal. The signal points constellation are a subset of the coordinates of the unity circle.

We conclude, that the maximum theoretical possible gain equals:

\[
\text{gain} = 10 \cdot 10^{\log 10/4} = 4 \text{ dB}
\]

and the lower bound for the error probability (3) equals:

\[
P_b \geq N(d_{\text{free}}) \cdot Q\left(\sqrt{10}/(2\sigma)\right) = Q\left(\sqrt{10}/(2\sigma)\right)
\]

where \(N(d_{\text{free}})\) in this situation means the number of paths having a distance of \(d_{\text{free}}\) to the all-zero path.

The codulation with 8PSK modulation.

The codulation system considered in this section is given in figure 3.6.

Figure 3.6 The codulation system with 8PSK modulation.
The convolutional encoder chosen for is an eight-state machine, because with an encoder having less stages (4) it is not possible to fulfill the assignment requirements mentioned in section 3.1. It then is not possible to assign the four subsets to the four state and simultaneously fulfill requirement 1, unless all the transitions are parallel transitions, which is undesirable. For this reason we tried an eight-state encoder. The used code is a systematic code with rate \( R = \frac{1}{2} \) and having an optimum distance profile of six. The encoder consists of three delay elements, which delay the input over one bit time each.

All possible transitions from a state \( s_{10} \) at time \( t_1 \) to a succeeding state \( s_{210} \) at time \( t_2 \) (\( t_1 < t_2 \)) are given in the trellis diagram of figure 3.7, together with the outputs \( z_{210} \) during the transitions.

![Trellis Diagram](image)

*Figure 3.7 The convolutional encoder and the trellis diagram.*
From the trellis diagram we notice, that even with an eight state encoder it is not possible to fulfill the requirements. The thick printed path, merging after four transitions in state 000, originating from state 001, receives signals from two different subsets. To solve this problem the number of stages must at least be doubled. This however introduces mapping problems (too complex) and moreover the complexity of the Viterbi decoder will be even more than twice the complexity of the configuration with an eight-state encoder. So we accept the present configuration, knowing that it is not optimal.

The signal points are now again assigned to the transitions in the most optimal way to assure maximum possible free distance. The set partitioning of the 8PSK signal point constellation is given in figure 3.8.

![Figure 3.8 Partitioning of the 8PSK signal points.](image)

We let the output values $z_0$ and $z_1$ select the subsets and $z_2$ select the signal point in the subset.

We assume again that the all-zero sequence is transmitted. The path differing from the all-zero sequence in as less as possible positions is printed thick in the trellis diagram of figure 3.7. The distance from the
all-zero path is:
\[ d_{\text{free}} = \sqrt{d^2(sp\ 0, sp\ 4) + d^2(sp\ 0, sp\ 3) + d^2(sp\ 0, sp\ 1) + d^2(sp\ 0, sp\ 3)} \]

\[ = \sqrt{(4 + 2(2+\sqrt{2}) + 0.765)} = 3.38 \]

The gain of the codulation system versus uncoded BPSK modulation can be calculated by means of equation (5), where \( E_{sp,\text{coded}} \) is unaffected for the same reason as mentioned in the previous section:

\[ \text{gain} = 10 \cdot \log\left(\frac{11.4}{4}\right) = 4.55 \text{ dB} \]

The lower bound for the error probability of coded 8PSK is (3):

\[ P_{b,\text{8PSK}} \geq Q\left(\sqrt{11.4/(2\sigma)}\right) \quad \text{where } N(d_{\text{free}}) = 1 \]

### 3.2.2 Codulation system with 9PSK modulation versus uncoded 3PSK modulation

The reference and codulation systems considered in this section are shown in figure 3.9.

The 3PSK modulation system has an error performance worse than the BPSK
system (about 0.6 dB). However the necessary bandwidth is smaller. The complexity of the system is major to that of BPSK or QPSK, because you need 3 carriers in phase rotated over 120°, which is hard to realize with the necessary stability. In spite of this we choose it as a reference system and will relate the results to the BPSK system.

The convolutional encoder is somewhat different from the previous ones, because it has to operate with three level symbols (trits) instead of bits. The signal points constellation of 3PSK is not a power of two, neither for 6PSK or 9PSK. This results in the impossibility of applying the TCM theory according to Ungerboeck, and finding a proper convolutional encoder for doubling the signal set. If we however introduce the operation with trits, it can be done quite simple. 9PSK has $3^2$ signal points and 3PSK has $3^1$ signal points. We conclude, that by using a convolutional encoder of rate $R_{\text{trits}} = 1/2$, we almost solved the problem. We take the convolutional encoder of figure 3.4 and modify it for operation with three logic levels, see figure 3.10.

**figure 3.10** The convolutional encoder of rate $R = 1/2$, operating with three logic levels.

Clarification of figure 3.10:

- The trit converter (1) converts an incoming symbol (a trit) in a two bit word, according to the following procedure:
"0" is converted to 00,
"1" is converted to 01,
"2" is converted to 10.

-The memory elements (2) containing two bits, related to one trit value, delayed over 1 symbol (trit) interval.
-The adder operates modulo-3 instead of modulo-2 in the previous configurations.
-The bit converter (3) converts a two bit word to one trit.

Since each of the two memories can contain three different trit values, the encoder consists of nine states. From each state three possible transitions are possible, each related to one of the three possible logic levels. The trellis diagram for the encoder is shown in figure 3.11. Together with each transition the corresponding output values are given. For clarity they are given in decimal form.

![Trellis diagram of the convolutional encoder](image)

*figure 3.11 The trellis diagram of the convolutional encoder.*
Again set partitioning is used to assign the signal points to the transitions and optimize the maximum free distance. The partitioning of the 9PSK signal point constellation is shown in figure 3.12.

\[ d_{\text{free}} = \sqrt{(d^2(\text{sp } 0, \text{sp } 3) + d^2(\text{sp } 0, \text{sp } 1) + d^2(\text{sp } 0, \text{sp } 6))} \]
\[ = \sqrt{(2 \cdot \sqrt{3}^2 + (0.68)^2)} = 2.54 \]

The gain over the uncoded 3PSK system can be calculated according to (5), where \( E_{sp} \) is equal for both situations. The free distance of the uncoded 3PSK system \( d_0 = \sqrt{3} \).

\[
\text{gain} = 10^{10 \log (6.47/3)} = 3.34 \text{ dB}
\]

The lower bound for the error probability of coded 9PSK is (3):

\[
P_b = Q(2.54/(2\sigma)) \quad \text{where } N(d_{\text{free}}) = 1
\]
The gain over uncoded BPSK according to (5):

\[
gain = 10^{10 \log (6.47/4)} = 2.01 \text{ dB}
\]

Table 2 gives a summary of the results in terms of gain (assuming unlimited coding and decoding effort), bandwidth requirements and error performance versus uncoded BPSK modulation.

Table 2.

<table>
<thead>
<tr>
<th>modem</th>
<th>rate R</th>
<th>BT</th>
<th>gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>QPSK</td>
<td>1/2</td>
<td>2rb</td>
<td>4.0 dB</td>
</tr>
<tr>
<td>8PSK</td>
<td>1/3</td>
<td>2rb</td>
<td>4.55 dB</td>
</tr>
<tr>
<td>9PSK</td>
<td>1/2</td>
<td>≈ 1.6rs</td>
<td>2.01 dB</td>
</tr>
<tr>
<td>3PSK</td>
<td>X</td>
<td>≈ 1.6rs</td>
<td>-1.25 dB</td>
</tr>
</tbody>
</table>

From Table 2 we conclude, that in terms of gain versus uncoded BPSK modulation, the codulation system using 8PSK is superior, but only 0.6 dB better than the codulation system using QPSK modulation.

3.2.3 The complexity.

In this section we will briefly address the aspects of complexity of the codulation systems using QPSK respectively 8PSK modulation. The complexity of both systems will determine if it is worthwhile to implement the 8PSK instead of the QPSK system to gain about 0.6 dB more. From block diagrams of both systems in reference [7] we conclude, that the complexity of the used modulation system increases rapidly with the number of phases. If we compare the block diagrams of both systems, we can conclude that both modulator and demodulator for the 8PSK system are far more complex than for QPSK. Besides this fact, another important aspect is the fact, that the encoder of the 8PSK codulation system is an
eight state machine, while the encoder for the QPSK codulation system only has four states. For the soft-decision Viterbi decoder, this means an increase in complexity by at least a factor 2. Therefore we think, that in this case the codulation system using QPSK modulation, is the best solution in terms of complexity, costs, bandwidth requirements and power consumption to implement for a rural communication system transmitting pseudo-ternary signals.
CHAPTER 4.

The implementation of a 4-level soft-decision Viterbi decoder, operating at an encoded bit rate of 4.096 Mbit/s.

4.1. Model of the communication system.

The communication system considered in this chapter is shown in figure 2.7, which gives a better insight in the environment in which the Viterbi decoder is used. The pseudo-ternary input data are first of all converted to a binary data sequence, for reasons mentioned earlier in chapter 2. The binary data are input for a convolutional encoder which add some code information. Then a binary to one-of-four phases mapping provides the code with an optimum free Euclidean distance. Next the mapped convolutionally encoded data modulate the carrier of a QPSK modulator.

At the receive side the output signal of the QPSK demodulator, including additive white Gaussian noise is coherently detected. The now obtained so called baseband signal is sampled by the sampling section, which converts the demodulated "analog" signals into digital words of two bits ($2^2 = 4$ quantization levels). Finally the encoded data sequence is decoded by the Viterbi decoder.

4.2. The general structure of the decoder, including the A/D converter.

4.2.1. The encoder.

The mechanism of the Viterbi decoder is very much related to the used convolutional encoder. Because of this we give a short description of the used convolutional encoder. The four state convolutional encoder considered in this chapter, whose performance is described in [1], differs in configuration with the one used in chapter 2. But as already mentioned earlier, this can be done without any loss of generality. Since both convolutional encoders can have the same properties. The reason for this is, that the performance of the used encoder, in relation with the
whole communication system, not only depends on the configuration of the modulo-2 adders, but also on the mapping function. We therefore can conclude, that both of these encoders, having the same constraint length, will provide an equal gain in signal to noise ratio if we adapt the mapping function of the second one.

Then encoder consists of one shift register containing three stages and two modulo-2 adders. The three stages assume a total of $2^3 = 8$ conditions. However the third bit is discarded every time a new data bit enters the shift register. Consequently the state of the encoder is determined by the two most recent data bits and the number of states is limited to $2^2 = 4$.

The rate of the encoder is $1/2$ ($R = 1/2$) and the free distance is five ($d_{\text{free}} = 5$), which defines the error correcting properties of the used code.

The mapping function of the encoder is given in figure 4.2. and can be easily derived from the trellis diagram.

*figure 4.1 The convolutional encoder.*
4.2. The Viterbi decoder.

By means of the block diagram of appendix A some rough calculations were made in order to get a first look at the speed limitations of the decoder in relation to the used digital logic family (e.g. CMOS, TTL, ECL ect.). We came to the conclusion that by using ECL-logic and processing two bits at a time, a maximum data rate of about 30 Mbit/s should theoretically be possible. The use of low power Schottky TTL logic (74LS..) respectively
fast TTL logic (74F...) limits the maximum bit rate to about 2 Mbit/s respectively 20 Mbit/s, if we also process two bits at a time. Notice, that the encoder with rate $R = 1/2$ doubles the bit rate of the input signal to 4.096 Mbit/s.

To ensure a minimum bit rate of 4.096 Mbit/s and to provide the hardware realization with some margins, we choose for the implementation of the Viterbi decoder in fast TTL logic, which can work at a maximum theoretical data rate of 20 Mbit/s.

The decoder is a soft-decision maximum-likelihood Viterbi decoder with four quantization levels. Because of the rather major complexity of the decoder, we divide the system in six significant blocks, each having its own specific function. By doing this we were able to test and if necessary modify the blocks separately. The block diagram of the decoder, including the A/D converter is shown in appendix A and is built up by the following functional blocks:

1. Sampling section,
2. Branch metric computation,
3. State metric computation,
4. Survivor path memory,
5. Output selection unit,
6. The clock control section.

All these blocks mentioned above will be discussed in the following six sections.

The sampling section.

In order to be able to process digitally the symmetric output signal of the QPSK demodulator, we have to convert it into a binary word. We realize this by quantizing the analog signal to 4 threshold levels, which implies a binary word of two bits for each sampled value. The spacing of the thresholds is predefined and according to reference [3] nearly optimal if equally spaced. For a four level soft-decision decoder, the normalized spacing value $T \{ = \frac{|b_1 - b_{i-1}|}{\sqrt{E_s}} \}$ is chosen to be 0.5.
which gives nearly optimum bit error rate performance.

![Threshold Spacing](image)

**Figure 4.4** The threshold spacing.

An encoded information bit is converted by the convolutional encoder of rate $R = 1/2$, to a binary word of two bits. At the demodulator these two bits are sampled by the sampling section and converted to a two bit word each. This means, that we have to process two data (four soft-decision) bits at a time in order to be able to decode according to the Viterbi algorithm, since this uses the structure of the trellis diagram in which each transition between two successive states is related to two hard-decision data bits. For this reason it is necessary to process two incoming data words of two bits each at a time. For the converter itself, this means that it must consists of a set of two sampling sections as shown in the block diagram of figure 4.5.

![Block Diagram](image)

**Figure 4.5** The block diagram of the sampling section.

The "analog" input signal of the converter is compared to three thresholds. The A/D converter consists of three high speed comparators with TTL output. The output of the two sets of comparators is delivered to an encoder, which task is the conversion of the six (3 per converter) incoming signals into two code words of length two.
To prevent data conflicts during the further data processing, the output signals of the encoder are stored in a latch, which is accomplished with four D-flipflops. A detailed diagram of the sampling section can be found in appendix Bl. An RC network forces the system into a known state every time it is started up.

The branch metric calculation (BMC).

Each branch metric is a measure of the correlation between the corresponding received code symbol from the sampling section and the set of four possible data words of the four possible transitions of the trellis diagram \(D = \{00, 01, 10, 11\}\). For hard decision, this operation can quite easily be accomplished by just computing the Hamming distance. For 4-level soft decision this is somewhat more difficult. However it is still possible to compute a measure of correlation, with a little more effort. From figure 4.4, we see that the binary word 00 corresponds to a received 0 (logic low level) and that the binary word 11 corresponds to a received 1 (logic high level). The task of the BMC section is to compare the two received code symbols with the transition symbol set \(D\) and to compute a distance measure. For soft decision and four quantization levels \(D\) changes to \(D' = \{00 \ 00, \ 00 \ 11, \ 11 \ 00, \ 11 \ 11\}\). Now we are in the position to derive a procedure to compute the branch metric. This procedure consists of two steps. The first step is to compare the set of code symbols \(C = \{s_1, s_2\}\) with the elements of set \(B = \{00, 11\}\). The amount of correlation between both sets has to be expressed by a binary value. For the first element of set \(B, 00\), this can easily be accomplished by adding (modulo-2) \(s_1\) resp. \(s_2\) to the symbol 00. By doing so the maximum possible distance related to 00 (=minimum correlation) is given by 11 if \(s_1\) resp. \(s_2\) are 11. In other words, the received code symbol is the metric. For the symbol 11 we first have to invert \(s_1\) resp. \(s_2\), before adding \(s_1\) resp. \(s_2\) to 11. Now the maximum possible distance related to 11 is also 11. In other words, the received code symbol is the inverted metric. We now have the so called pseudo distance metrics for \(s_1\) and \(s_2\). To form the four overall branch metrics we have to sum both metrics in four different ways as shown in the block diagram of figure 4.6, which is the second and final step of the procedure.
The branch metric calculation.

A detailed diagram of the branch metric calculation section can be found in appendix B2.

The state metric calculation (SMC).

The state metric (the smallest path metric) of state s2 is the summation of the state metric of the previous state s2 at the beginning of the branch which merges in state s2 and the branch metric of that particular branch. Because of the fact that we have four encoder states, we also must have four state metric computation sections, one for each encoder state.

Another task of this section is to compare both possible path metrics for each state and to select and store the smallest one, for the next state metric updating cycle. For this reason this section is often called in literature the add-compare-select section (ACS). The block diagram of the ACS is shown in figure 4.7.
Designing the hardware for the metric updating process requires trade-offs among the following variables:

1. sufficiently fine input quantization,
2. path metric size with available logic,
3. maximum data rate.

The computation of the branch metric is less time critical than the rest of the state metric computation. This because of the fact that the branch metric may be calculated and stored for further processing. The state metric updating however must be accomplished within a half clock period. Since each path metric calculation requires one of the previous state metrics. The entire processing must be completed within one clock period in order for the state metric to be available for the next calculation. Here we already see where lies the most critical problem in designing a Viterbi decoder.

The repetitive structure of the state metric updating process results in a second design problem. Because the state metric is a monotonically increasing function, the repetitive structure results in an infinitely growth. In order to be able to implement this function we have to rescale the state metric from time to time. The optimum way of doing this is to subtract after each state metric updating cycle the smallest metric from all the other metrics. This however is a quite difficult to implement problem and above that also time consuming. An other nearly optimum way of rescaling is to subtract a predefined value from all the metrics,
every time a metric reaches the threshold value.

Since rescaling is used to prevent metric overflow, the number of bits required for each path (state) metric is determined by the maximum variation among the state metrics as well as by the normalization process, shown in the next section. Fortunately the maximum variation among the state metrics is bounded as, will be shown in the following discussion.

Let K be the constraint length defined by the used convolutional encoder in symbols (K = 3). Let $B_{\text{max}}$ be the maximum value of the branch metric and let $P_{\text{M}}(t,s)$ be the value of the path metric of state $s$ ($s \in S$) at time $t$ ($t \in T$). $P_{\text{min}}$ and $P_{\text{max}}$ are the minimum respectively maximum value of the path metric. The threshold value is given by $M$. Assume $M$ to be chosen as $M > B_{\text{max}}$. In the beginning of the transmission the minimum value of the path metric $P_{\text{min}}$ can be bounded as shown in the following relation:

$$0 \leq P_{\text{min}} < M \quad (1)$$

Consider next the case in which the minimum value of the path metric is less than $M$ at time $t-1$ and exceeds the threshold during the next transition. This makes the minimum value more or equal to $M$:

$$0 \leq P_{\text{min}} \leq M \quad (2)$$

$$M \leq P_{\text{M}}(s,t) < M + B_{\text{max}} \quad (3)$$

In this case the maximum value of the path metric value is bounded by:

$$P_{\text{M}}(s,t) < M + B_{\text{max}} \quad (4)$$

This can be proved by using the property, that the spread of $\Delta P_{\text{M}}$ is bounded as [8]:

$$\Delta P_{\text{M}} = P_{\text{max}} - P_{\text{min}} \leq (K-1)B_{\text{max}} \quad (5)$$
Equation (3) and (4) mean that all the path (state) metric values are bounded by \((M + K.BM_{\text{max}})\) at the moment that \(PM(t)\) exceeds the threshold \(M\). On this same moment however we subtract \(M\) from each value \(PM(s,t)\). This leads to the following relation:

\[
0 \leq PM_{\text{min}}(t) < BM_{\text{max}} < M \quad (6)
\]

\[
PM_{\text{max}}(t) < K.BM_{\text{max}} \quad (7)
\]

From the preceding we can conclude, that the number of bits assigned to the path (state) metrics must permit differences as large as \(K.BM_{\text{max}}\).

The threshold value \(M\) and the metric size.

The maximum variation among the path metrics \(\Delta PM = BM_{\text{max}} \times K = 6 \times 3 = 18\), which makes a minimum path metric size of 5 bits, implying the need for two 4 bit adders (74F283). Now we are able to determine the optimal value for \(M\). The minimum and maximum value of \(PM_{\text{max}}(t)\) starting the rescaling process and giving the maximum and minimum value of \(PM_{\text{min}}(t)\) are given by:

\[
PM_{\text{max}} = 2^6 - 1 + BM_{\text{max}} = 63 + 6 = 69
\]

\[
PM_{\text{min}} = 69 - \Delta PM = 69 - (3-1).6 = 57
\]

\[
PM_{\text{max}} = 58 - BM_{\text{max}} = 58 + 6 = 64
\]

\[
PM_{\text{min}} = 64 - \Delta PM = 64 - 12 = 52
\]

resulting in the following relation, giving the possible path metric values before rescaling has taken place:

\[
52 \leq PM(s,t) \leq 69 \quad (8)
\]

So subtracting a maximum value of 52 is possible when one of the metrics exceeds the threshold limit of 53. Now it's easy to chose a nearly optimum value for \(M\). If we chose \(M\) to be the largest possible power of
two smaller than 52. This results in a value for \( m \) of \( 2^5 = 32 \). If we now choose the metric size to be 6 bits, the rescaling procedure only consists in detecting whether bit 7 of one (or more) of the path metrics is (are) logic "1" and inverting bit 6 of all the metrics if this is the case. This has the same effect as subtracting 32 of all the path metrics, but much more time efficient.

In summary, the rescaling procedure has the following features:

1. It is not necessary to find the minimum path metric after each updating procedure,
2. Subtraction has not to be executed after each updating procedure, but only occasionally,
3. We were able to speed up the updating procedure by choosing the threshold equal to a power of two. Subtraction then can be accomplished by bit inversion of the path metric.

For a detailed diagram of the ACS see appendix B3.

The survivor path memory.

Associated with each state metric is a kind of shift register, storing that sequence of information bits along the decoding process, corresponding to the smallest state metric. These four shift registers, one for each state metric, contain the most likely decoded data sequences leading to each state. The updating process of these registers is executed parallel with the path metric computation.

If we review again the trellis diagram of figure 3.11 on page 31, we see that e.g. state 00 can only be reached from a preceding state, if the most recent decoder input bit is a 0. The same can be said for state 01. For the states 10 and 11 the most recent decoder input bit ought to be a logic 1.

The updating procedure is illustrated for state 00, to make clear the basic features of the survivor sequence memory. Suppose that \( \text{PM}_1 < \text{PM}_2 \) (path metric for upper resp. under transition). Then the contents of
register 1 is simply updated by shifting all data one unit to the right and at the same moment shifting in a zero bit at the first (most left stage) stage of the register. If however \( PM1 > PM2 \) the contents of register 1 is updated by parallel shifting into register 1 the contents of register 2, corresponding to state 01, since that path is then the most likely one of being sent. Simultaneously a 0 is shifted in at the first stage. This same procedure holds for state 01, except that register 1 changes to register 3 (state 10) and register 2 changes to register 4 (state 11). For the states 10 and 11 something likewise can be told, except that we must shift in a 1 in the first stage.

To select the appropriate source of the survivor path bits, each stage of the shift registers is preceded by a 2-input multiplexer, selecting the data, as shown in the block diagram of figure 4.8 [9].

![Figure 4.8 The block diagram of the path memory.](image)

The output (select) signals of the comparators of the ACS section are the control signals for the 2-input multiplexers. Since the output of the first two stages is independent of the control or the input signals, they can be omitted for implementation.

It has been found through computer simulations [3], that a memory length of about five times the constraint length for each state, will perform very well, since it is highly probable that each of the four surviving paths have diverged from one common state not further back than approximately four or five times the constraint length. The final stage of each register may then be selected to determine the most likely information bit being transmitted five bit intervals ago. To improve the performance we select the contents of the final stage of that shift
register, belonging to the most likely state, having the smallest momentary overall path metric. This is accomplished by the select unit discussed in the next section.

The path memory consists of sixteen 74F298, which is an IC with four memory elements, preceded by a 2-input multiplexers and very well suited for our purpose. For a detailed diagram of the survivor path memory see appendix B4.

The output selection unit (OS).

The output selection unit compares, after each state metric updating cycle, the state metrics of the four possible states and selects the one with the smallest state metric. The output, of the to this particular state related survivor path register, is then chosen as the one with the highest probability of being transmitted. The selection procedure consists of two steps. During the first step the state metrics of the states 00 and 01 are compared and the smallest one is the input for the comparison during the second step. The same holds for the states 10 and 11. Then during the second step both smallest metrics, obtained from the first step, are compared, resulting in the smallest overall state metric belonging to, let us say state 00. With this information the OS unit chooses the contents of the final stage of the to this state related survivor path memory. The block diagram of the OS unit is shown in figure 4.9.

![Block Diagram of the Output Selection Unit](image)

Figure 4.9 The block diagram of the output selection unit.

46
The three output signals of the comparators are delivered to an 8-input multiplexer, which is modified for our purpose to an 3-input multiplexer. The output of the multiplexer is clocked in a D-flipflop, to make it stable and avoid any undefined output signals during the succeeding updating cycle. For a detailed diagram of the OS unit see appendix B5.

The clock control section.

Simultaneously with every data bit, the QPSK demodulator generates a clock pulse with half the width of the bit interval. For our purpose we invert the clock pulses for reasons that will be explained later on in this section. The five functional blocks of which the Viterbi decoder consists are all edge triggered to make it possible to execute two blocks at a time. This will be shown by means of the block diagram of figure 4.10.

![Clock Configuration Diagram](image)

*figure 4.10 Timing clock configuration diagram of the Viterbi decoder.*
During the first transition of the clock pulse (the rising edge 1), the two by the sampling section quantized analog input signals are latched and stable for the computation of the branch and state metrics. The second transition of the clock signal is the starting signal for the ACS section to latch the new state metrics computed during the time interval \((t_2 - t_1)\) and make it stable for the final step of this updating cycle performed by the OS unit. During the third transition, which is the same as the first, the decoded data bit is clocked into a D-flipflop to avoid any undefined output levels during the next state metric updating cycle.

The reason for inverting the clock pulse is now quite easy to understand. If we, instead of first inverting the clock signal, deliver it straight to the several block of the decoder, the state metric updating cycle would compute the path metrics with undefined values for the branch metrics (sampling of the analog demodulator output would happen on the falling edge of the data bit). This can be solved by delaying the clock pulse over the propagation delay time of the sampling section. However, by just inverting the clock pulse, we manage to create a time delay margin of about half the clock interval.

Next we consider the maximum possible data rate that may be applied to the Viterbi decoder. From figure 4.9 we conclude that within the time interval \((t_2 - t_1)\), the computation of the branch metric and the state metric has to be executed and completed to guarantee proper operation of the decoder. Now it is possible to give a good estimation the maximum possible data rate if we make use of table 1. This table 1 gives the several propagation delay times of all the functional block of the Viterbi decoder.
Table 1. The propagation delay times (in ns).

<table>
<thead>
<tr>
<th>functional block</th>
<th>delay time (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>A/D conv.</td>
<td>24</td>
</tr>
<tr>
<td>BMC section</td>
<td>17.5</td>
</tr>
<tr>
<td>ACS section</td>
<td>45</td>
</tr>
<tr>
<td>path memory</td>
<td>26</td>
</tr>
<tr>
<td>output section</td>
<td>56</td>
</tr>
</tbody>
</table>

The propagation delay of the BMC section and the ACS section is equal to 17.5ns + 40ns = 60ns (guard margins of 2.5ns). If we compare these with the propagation delay times of the other functional blocks, it is clear, that the combination of the BMC and the ACS section defines the maximum encoded data rate. Since the state metric updating cycle has the largest propagation delay time of all and must be completed in a half clock period \((t_2 - t_1)\).

The maximum encoded data rate:

\[
\frac{2}{2.(t_2 - t_1)} = \frac{1}{60E-9} = 16.6 \text{ Mbit/sec.}
\]

The factor two in the counter appears because of the fact, that we process two encoded bits in one bit interval.

The test procedure of the decoder, discussed in the next chapter, has been done by means of baseband signals. This leads again to a timing problem. The D-flipflop introduces a delay of the clock pulses related to the data bits, of approximately 8.5ns. In principle this would not be any problem, but it is more secure to sample the demodulator output signal on half the bit time. For this reason we delay the clock pulses with the same amount of time by means of a TTL nand gate (propagation delay approx. 9 ns).
The theoretically expected BER performance.

The structure of the Viterbi decoder is now exactly known and it is possible to calculate the BER performance by means of the equations derived in chapter 3. The only variables that still have to be defined are the truncation values for k and n. Where k is the distance between the correct and incorrect path. Because the fact that we use an encoder with a free distance of 5, the minimum value for k is 5. The maximum value for k is bounded by the length of the survivor path memory, because this is exactly the maximum possible positions in which two paths can differ and still be noticed by the decoder. So the maximum value for k is 16, the length of the survivor path memory. To obtain a tight lower bound for the value of n, we rewrite (7) in chapter 3 as follows [2]:

\[ q_k(n) = \Pr\left( \sum_{j=1}^{k} ((Q + 1) - 2 \cdot j) = n \right) \]  

(10)

where \( i \in \{1, 2, \ldots, Q\} \) means the decision region \( i \) in which the \( j \)-th received signal level \( r_j \) (chapter 2 (1)) is contained and \( Q \) the number of quantization levels. The minimum value of relation (10) is obtained if
for every possible value of $j$, $1_j = Q = 4$. This results in a minimum value of $n$ related to $k$ of:

$$k - 3 \leq n$$

(11)

In summary:

$$5 \leq k \leq 16$$

(12)

$$-3k \leq n \leq -1$$

(13)

With the bounded values of $k$ and $n$ shown in (12) and (13) we developed a Pascal computer program, which computes the theoretical bit error rate of a soft-decision Viterbi decoder with four equal spaced thresholds and a survivor path memory of length 16. The only input parameter that is needed is the ratio $E_b/N_0$ in dB.

The results for several significant input values are given in the graphs of figure 4.12, together with the BER performance of uncoded QPSK modulation.

BER performance of 4-level soft decision
Viterbi decoder with $R=0.5$ and $K=3$

![Graph of the theoretical BER performance of the realized soft-decision Viterbi decoder.](image)

Figure 4.12 Graph of the theoretical BER performance of the realized soft-decision Viterbi decoder.

For a listing of the Pascal program see appendix C1.
CHAPTER 5.

Testing procedure and overall conclusions.

5.1 Testing procedure.

The testing procedure is split up into two phases. The first phase only consists of verifying the branch and state metric values for the four possible states with a known predefined value for the inputs (S1, S2) of the branch metric calculation section (appendix B2). Therefore these results are independent of the characteristics of the A/D conversion, but give a good indication of the so called "static" behavior of the decoder. During the second phase more realistic measurements (e.g. error probability as function of $E_s/N_0$) are made by means of a pseudo-random data generator an error detector and two noise generators to disturb the two encoded data sequences. But first of all we consider phase one, in which we make some "static" measurements.

First phase.

We developed a pascal computer program (appendix C2), which simulates the BMC and the ACS sections for a given input sequence for S1 and S2 (appendix B2) and calculates all possible branch and state metric values together with the four select signals (appendix B3) for the first 100 clock pulses.

The metric values computed by the computer program equaled the measured ones for clock pulses with a pulse width of at least 70ns. This results in a maximum possible encoded bit rate for these sections, of approximately 14 Mbit/s. This is about 2.5 Mbit/s less than the one calculated in chapter 4, which can be explained by the fact, that the rise and fall times of the clock and data pulses are not infinitely small.
Second phase.

In chapter 2 we derived a relation for the upper bound of the bit error rate performance of the 4-level soft-decision Viterbi decoder as a function of the ratio $E_b/N_0$. In this section we want to compare the analytically derived bit error rate performance with the measured one, obtained by means of the system setup shown in figure 5.1. This system differs from the real system (figure 2.7) by the absent of a QPSK modulator and demodulator. This because of the fact, that they were not available at the time the measurements took place.

![Diagram of system setup for measuring the BER of the Viterbi decoder](image)

**Figure 5.1** The system setup for measuring the BER of the Viterbi decoder.

As shown in the figure above we only use baseband signals. This introduces two problems. First, because the output of the convolutional encoder is a TTL signal instead of a symmetric signal in which the logic "high level" equals a positive voltage $V$ and the logic low level equals the same voltage $V$ but negative.

To solve this minor problem we make use of two (one for each channel) series connection of two CMOS nand gates (HEF4011) of which the positive power supply is connected to $+5$ Volt and the ground to $-5$ Volt. By doing this we realized a level shifter, which maps the TTL output signals of the encoder to about $+5$ and $-5$ Volts. This level shifter operates
well for bit rates up to approximately 10 Mbit/s.

Second, the derived equations for the BER performance were based on the communication system of figure 2.7 (called system 1), where the input of the Viterbi decoder is a single serial data sequence. In the system setup (called system 2) however the input of the Viterbi decoder consists of two data sequences at a signal rate of half the one in system 1 at the output of the demodulator. The question is, if it is possible to measure the SNR and the BER by means of system 2 and next compare the results with the computed upper bound of the BER performance.

For the measurements we increase the information rate to \( r_b = 2.5 \text{ Mbit/s} \) and use two uncorrelated white noise generators, band limited to \( B = 5 \text{ MHz} \) and each having equal average noise power, the measured SNR in system 1 is the same as the SNR of system 2, if the average noise power per channel for system 2 is the same as the noise power at the output of the demodulator in system 1.

Proof:

The SNR of system 1 in figure 2.7 is given by:

\[
\frac{S_1}{N_1} = \frac{E_s}{N_0} \cdot \frac{2r_b}{B}\]

where

\( E_s \) = the signal energy per coded symbol,
\( r_b \) = the bit or information rate at the input of the encoder,
\( B \) = the bandwidth of the channel.

For system 2 we can write for the SNR:

\[
\frac{S_2}{N_2} = \frac{E_{s_2}}{N_0} \cdot \frac{r_b}{B} \quad \text{for both data sequences in system 2.}
\]

Because of the fact, that \( E_{s_2} = 2 \ E_{s_1} \), we can conclude that for both
systems, the signal-to-noise ratios are equal and that for system 2 the relation between $E_{s2}$ and $E_b$ is given by:

$$E_{s2} = E_b$$

For system 1 this relation is given by:

$$E_{s1} = 0.5 E_b$$

In summary, the measured value for the SNR in the system setup is equal to $E_{s1}/N_0$, because the bit rate $r_b = 2.5 \text{ Mbit/s}$ and $B = 2. r_b$. In order to get the ratio $E_b/N_0$ (in dB) we only have to add 3 dB to the ratio $E_{s1}/N_0$.

The used noise generators are power limited and not powerful enough to disturb a voltage level of $\pm 5 \text{ volt}$. We therefore reduce the voltage level to $\pm 0.5 \text{ volt}$ by means of two in dB calibrated attenuators, one for each encoded bit sequence.

5.1.2. Results and conclusions.

The input of the convolutional encoder is a pseudo-random NRZ bit sequence. This results in the fact, that the measured average signal energy per coded bit is constant in time and only has to be measured once. Thus the only parameter left is the noise power. By means of a rms voltage meter we determine $\sqrt{E_{s1}^2}$ and $\sqrt{N_0^2}$ and next calculate the $20 \cdot 10 \log(E_{s1}/N_0)$. By adding 3 dB to these values we finally get the desired ratio of $E_b/N_0$, as shown in table 3.
Table 3.

<table>
<thead>
<tr>
<th>$E_s/N_0$ in dB</th>
<th>$E_b/N_0$ in dB</th>
<th>$P_B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.36</td>
<td>9.36</td>
<td>xxxxxxxx</td>
</tr>
<tr>
<td>5.68</td>
<td>8.38</td>
<td>1.0 $10^{-8}$</td>
</tr>
<tr>
<td>5.03</td>
<td>8.03</td>
<td>1.7 $10^{-7}$</td>
</tr>
<tr>
<td>4.35</td>
<td>7.35</td>
<td>3.3 $10^{-7}$</td>
</tr>
<tr>
<td>3.69</td>
<td>6.69</td>
<td>2.5 $10^{-6}$</td>
</tr>
<tr>
<td>3.27</td>
<td>6.27</td>
<td>1.0 $10^{-5}$</td>
</tr>
<tr>
<td>2.38</td>
<td>5.38</td>
<td>2.1 $10^{-5}$</td>
</tr>
<tr>
<td>2.37</td>
<td>5.37</td>
<td>4.5 $10^{-5}$</td>
</tr>
<tr>
<td>1.86</td>
<td>4.86</td>
<td>1.7 $10^{-4}$</td>
</tr>
<tr>
<td>1.37</td>
<td>4.37</td>
<td>3.7 $10^{-4}$</td>
</tr>
</tbody>
</table>

xxxxxxx = not measurable, error probability was too small.

The measured values of table 3 are plotted in figure 5.2 together with the graphs of the analytical derived upper bound for the BER performance of a 4-level soft-decision Viterbi decoder and the graphs of the BER performance of uncoded QPSK.
The graphics of the BER performance of uncoded QPSK and the measured and computed values for the 4-level soft decision Viterbi decoder.

From figure 5.2 we conclude, that the realized soft-decision Viterbi decoder has a bit-error-rate performance, which hardly differs from the analytical derived tight upper bound.

The maximum possible coding gain of the 4-level soft-decision Viterbi decoder is about 3.4 dB for small error probabilities ($P_b \leq 10^{-5}$), as can be concluded from figure 5.2. This also confirms the previously made remark, that the loss of information by using 4-level quantization instead of infinitely fine quantization is maximum 0.7 dB (in this case $4 - 3.4 = 0.6$ dB). We can conclude, that a codulation system using ideal QPSK modulation in combination with the realized coder/decoder part, has a coding gain versus uncoded BPSK modulation of approx. 3.3 dB, for small error probabilities ($P_b \leq 10^{-5}$).
CHAPTER 6.

6. The conclusions.

In the first part of this report we described an efficient way of transmitting a pseudo-ternary signal, coded according to the HDB-3 or 4B3T coding rule. The use of trellis-coded modulation, in combination with QPSK modulation (called codulation), turned out to be superior to most other modulation systems in terms of bandwidth utilization, information rate, power efficiency and implementation costs. The maximum possible gain for a codulation system using ideal QPSK modulation, a four stage convolutional encoder and a 4-level soft-decision Viterbi is 4 dB. This will be achieved under the assumption of unlimited coding and decoding effort and moreover infinitely fine quantization of the data sequence at the input of the Viterbi decoder.

In the second part of this report we discussed the development and realization in fast TTL (74F..) logic of a 4-level soft-decision Viterbi decoder operating at a minimum bit rate of 2.048 Mbit/s. In combination with a convolutional encoder having a constraint length $K = 3$ and a rate $R = 1/2$, this results in an encoded data rate at the input of the decoder of 4.096 Mbit/s. The maximum possible encoded data sequence, where the decoder can operate on, turned out to be about 14 Mbit/s, implying an unencoded data sequence at the input of the encoder of 7 Mbit/s. This however could not exactly be measured, since the logic of the encoder (74HC..) was the speed limiting factor. For information rates exceeding 5 Mbit/s, the encoder operated unreliable. But the so called "static" measurements in paragraph 5.1 confirmed, that the decoder still works reliable for an encoded data rate of 14 Mbit/s. This however is far more than the desired encoded bit rate of 4.096 Mbit/s.

The bit-error rate performance of the designed decoder is measured and turned out to be almost equal to the theoretical derived tight upper bound for the bit-error probability. Versus uncoded ideal QPSK modulation this results in a average coding gain of approx. 3.3 dB for small error probabilities. For this reason we think that the results are most satisfactorily.
ACKNOWLEDGEMENT.

At the end of this report I wish to thank the telecommunication division of the Eindhoven University of Technology for their support and for giving me the opportunity to accomplish my graduation work at this division. Especially I wish to thank my supervisor Ir. A.P. Verlijsdonk for his guidance and instructive discussions.
REFERENCES.

1. Viterbi A.J.
Convolutional codes and their performance in communication systems.

2. Yasuda Y., Hirata Y. and Ogawa A.
Bit error rate performance of soft decision Viterbi decoding.

3. Heller J.A.
Viterbi decoding for satellite and space communication.

4. Ungerboeck G.
Channel coding with multilevel phase signals.

5. Ungerboeck G.

6. Ungerboeck G.

7. Feher K.
Digital communications.
8. Fujino T., Moritani Y., Miyake M., Sakato Y. and Shiino H.
A 120 Mbit/s 8PSK modem with soft-decision Viterbi decoder.
IEEE, 7th International Conference on digital Satellite Communications.
May 1981.

Analog Viterbi decoding for high speed digital satellite channels.

10. Larsen K.J.
Short Convolutional codes with maximal free distance for rates 1/2, 1/3 and 1/4.

11. Wiggert D.
Error-control coding and applications.

12. Clark G.
Error-correcting coding for digital communication.

13. Lin S. and Costello D.
Error control coding.

14. Blahut R.E.
Theory and practice of error control codes.

15. Shanmugam K.S.
Digital and analog communication systems.
APPENDIX A Block diagram of the Viterbi decoder.
Appendix B1: Input section.

CLOCK

FROM OTHER A/D CONVERTOR

CONNECT PIN 8 TO VCC (14)
CLOCK INPUT
- CLOCK 1

U1A
74HC30
1
2
3

U1C
74HC04
9
10
11
12
13

U1D
74HC04
8

U2A
74HC04
1
2
3

ENCODER D-FLIPFLOP
- CLOCK 2

PATH MEMORY

ADD-COMPARE-SELECT SECTION

INPUT/OUTPUT SECTION

CLOCK DELAY AND INVERT SECTION
APPENDIX C1 Pascal program BER.

program prob;

const a=50;
   bk=16; {bound k-value}

type plotarray = array[1..a] of real;

var i : integer;
   k,kk,n,g : integer;
   EN,P1,P2,P3,P4 : real;
   PB : real;

   f:array[0..32] of real;

   PK:array[1..a] of real;
   QK:array[1..a] of real;
   q:plotarray;

procedure surch(11,12,13,14,kk:integer;Pl,P2,P3,P4:real;var q:plotarray);

var z,x : real;
   d1,d2,d3,d4 : real;

begin
   if 11>2 then begin
      for i:=1 to 11 do d1:=d1*P1;
     end;
   if 12>2 then begin
      for i:=1 to 12 do d2:=d2*P2;
     end;
   if 13>2 then begin
      for i:=1 to 13 do d3:=d3*P3;
     end;
   if 14>2 then begin
      for i:=1 to 14 do d4:=d4*P4;
     end;
   if 11<3 then d1:=exp(11*ln(P1));
   if 12<3 then d2:=exp(12*ln(P2));
   if 13<3 then d3:=exp(13*ln(P3));
if $14<3$ then $d_4 := \exp(14 \ln(P_4))$;

d_1 := d_2 + d_3 + d_4;


$q[ kk ] := (q[ kk ] + x / z)$;

end;

procedure simpson(a, b: real; var s: real);

const n = 50; \{ aantal stappen \}

var l, h, x, yy, d : real;

i, k : integer;

y : array[0..n] of real;

begin

l := a; h := (b - a) / n;

for i := 0 to n do begin

x := l; d := sqr(x);

yy := exp(-d);

y[i] := yy;

l := l + h;

end;

s := y[0] + y[n];

for i := 1 to 49 do begin

if $i / 2 <> \text{int}(i / 2)$ then k := 4
else k := 2;

s := s + k * y[1];

end;

s := s * h / 3 * l / sqrt(l);

end;

procedure wdk(k, kk, n: integer; var q: plotarray);

var 11, 12, 13, 14, z, s: integer;

111, 112, 113, 114: integer;

a, x : integer;

procedure find(11, 12, 13, 14, k, kk, n: integer; var

113, 114: integer; var q: plotarray);

begin

\[70\]
13:=13+1; 14:=14-1;

while 14<>kk do begin
    13:=13-1; 14:=14+1;
    if ((13-abs(13)=0) and (14-abs(14)=0)) then
        begin
            if (11+2*12+3*13+4*14) = (5*kk-n)/2 then
                begin
                    surch(11,12,13,14,kk,P1,P2,P3,P4,q);
                end;
            end;
        end;
    end;
end;

begin
    for a:=0 to k do begin
        11:=a;
        for x:=0 to (k-a) do begin
            13:=(k-a)-x; 14:=0;
            12:=x;
            find(11,12,13,14,k-a,kk,n,z,s,q);
        end;
    end;
end;

procedure transp(EN : real;var P1,P2,P3,P4 : real);{in dB}
const t=0.5; {threshold spacing normalized on Es}

var a,b,d,s : real;

begin
    EN:=sqrt(exp(EN/10*ln(10)));
    writeln(lst,'Es/NO ',EN);
    writeln(lst);
    a:=(t-1)-EN; b:=9;
    simpson(a, b, s);
    P1:=s;

    a:=-1*EN;
    simpson(a, b, s);d:=s;
    a:=(t-1)*EN;
    simpson(a, b, s);
    P2:=d-s;

    a:=(-1-t)*EN;
    simpson(a, b, s);d:=s;
    a:=-1*EN;
    simpson(a, b, s);
    P3:=d-s;

    71
\[ a = (1 + t) \cdot EN; \]
\[ \text{simpson}(a, b, s); \]
\[ P4 := s; \]
\[
\text{writeln}(\text{lst},'P1 = ',P1,' P2 = ',P2);
\]
\[
\text{writeln}(\text{lst},'P3 = ',P3,' P4 = ',P4);
\]
\text{end;}

\{ \text{MAIN PROGRAM} \}

\begin{verbatim}
\text{begin}
\text{ClrScr; for } i := 1 \text{ to } 32 \text{ do begin } \{k\}
\text{f}[0] := 1; \text{f}[1] := \text{f}[1-1] \times 1; \text{end;}

\text{for } i := 1 \text{ to } a \text{ do begin}
\text{q}[i] := 0; \text{PK}[i] := 0; \text{end;}

\text{write('give value for } Es/NO : '); \text{readln(EN); writeln; write\linebreak}
\text{writeln(lst);} \text{writeln(lst,' VALUE FOR } Es/NO : ');
\text{writeln(lst,' Eb/NO = ',EN+3); writeln(lst); transp(EN,P1,P2,P3,P4);

\text{for } k := 5 \text{ to } bk \text{ do begin}
\text{n := 0; kk := k; if } (n+k) \text{ mod } 2 = 0 \text{ then begin}
\text{wkd(k,kk,n,q);\{qk(n)\}} \text{end;}
\text{PK}[k] := 0.5 \times q[k] \times f[k];\{\text{voor vaste } k \text{ PK(1),PK(2),...} \}
\text{q}[k] := 0; \{\text{reset } q \text{ array}\}

\text{end;}
\text{for } g := 1 \text{ to } a \text{ do q}[k] := 0;
\text{for } k := 5 \text{ to } bk \text{ do begin}
\text{kk := k;}
\end{verbatim}
for n:= -3*k to -1 do begin
  if (n+k) mod 2=0 then
    begin
      wdk(k, kk, n, q);
    end;
end;

PK[k]:= q[k]*f[k];
end;

PB:= 0;
for k:= 5 to bk do begin
  PB:= PB+(k-4)*exp((k-5)*ln(2))*PK[k];
end;

writeln(lst, 'PB = ', PB);
end.
APPENDIX C2  Pascal program BMC + ACS.

program statemetric;

var a,a1,b,b1,c,c1,d,d1,e,f,g,h,x,y,i,z,n,s,t : integer;

aa,bb,cc,dd,tt : boolean;

ch : char;

fn : string[3];

s1 : array[1..100] of integer;

s2 : array[1..100] of integer;

bm00 : array[1..100] of integer;

bm01 : array[1..100] of integer;

bm10 : array[1..100] of integer;

bm11 : array[1..100] of integer;

inp : text;

begin

c1rscr;

begin
    writeln('1st');
    assign(inp,'seq.pas');
    reset(inp);
end;

for t:=1 to 100 do

begin
    readln(inp,s1[t],s2[t]);
end;

close(inp);

for t:=1 to 100 do

begin
    bm00[t]:=s1[t]+s2[t];
    bm01[t]:=s1[t]+(3-s2[t]);
    bm10[t]:=(3-s1[t])+s2[t];
    bm11[t]:=(3-s1[t])+(3-s2[t]);
    writeln('1st','bm00[t],'','bm01[t],'','bm10[t],'','bm11[t]',',t);
end;

a:=0; b:=0; c:=0; d:=0; i:=0;

x:=0;y:=0;
writeln(lst);
writeln(lst,' clock pulses SM 00 SM 01 SM 10 SM 11');
writeln(lst,'---------------------------------------------------------------
----------------');

for n:=1 to 3 do
begin
aa:=true;bb:=true;cc:=true;dd:=true;tt:=true;
t:=1;

while (aa and bb and cc and dd and tt) do
begin
  x := a + bm0O[t]; y := b + bm11[t];
  if x <= y then begin
    a1 := x; e:=0;
    end
  else begin
    a1 := y; e:=1;
    end;
  if a1<63 then aa:=true else aa:=false;

  x := c + bm10[t]; y := d + bm01[t];
  if x <= y then begin
    b1 := x; f:=0;
    end
  else begin
    b1 := y; f:=1;
    end;
  if b1<63 then bb:=true else bb:=false;

  x := a + bm11[t]; y := b + bm00[t];
  if x <= y then begin
    c1 := x; g:=0;
    end
  else begin
    c1 := y; g:=1;
    end;
  if c1<63 then cc:=true else cc:=false;

  x := c + bm01[t]; y := d + bm10[t];
  if x <= y then begin
    d1 := x; h:=0;
    end
  else begin
    d1 := y; h:=1;
    end;
end;
if \( d1<63 \) then \( dd:=true \) else \( dd:=false; \)

\( a:=a1;\) \( b:=b1;\) \( c:=c1;\) \( d:=d1;\)
\( i:=i\) +1; \( t:=t+1;\)
if \( t=99 \) then \( tt:=false; \)

if \( \text{not (aa and bb and cc and dd)} \) then
begin
\( a:=a-32;\) \( b:=b-32;\) \( c:=c-32;\) \( d:=d-32;\)
end;

begin
\( z:=i+3;\)
write(lst,'1'); write(lst,'a);
write(lst,'b);
write(lst,'c'); write(lst,'d);
write(lst,'e,f,g,h);
writeln(lst,'bmo0[t], bm01[t], bm10[t], bm11[t]);
end;
end;
end;
end.