# A Combined Equaliser and Decoder for Maximum Likelihood Decoding of Convolutional Codes in the presence of ISI. Incorporation into GSM 3GPP Standard.

*Panagiotis Mentis*

Der Fakultät für Elektrotechnik und Informationstechnik
der Universität der Bundeswehr München zur Erlangung
des akademischen Grades eines

Doktor-Ingenieur
(Dr.-Ing.)

vorgelegte Dissertation

München 27.10.2005

# Contents

# Abbreviations

| | |
|---|---|
| APP | A-Posteriori Probability |
| AWGN | Additive White Gaussian Noise |
| BER | Bit Error Rate |
| BPSK | Binary Phase-Shift Keying |
| BWT | Burrows-Wheeler Transform |
| DFA | Deterministic Finite Automaton |
| DFE | Decision Feedback Equaliser |
| FT | Fourier Transform |
| ISI | Intersymbol Interference |
| JPEG | Joint Photographic Experts Group |
| LAN | Local Area Network |
| LLR | Logarithm Likelihood Ratio |
| LZW | Lempel-Ziv-Welch |
| MAP | Maximum A-Posteriori |
| ML | Maximum Likelihood |
| MLSE | Maximum Likelihood Sequence Estimator |
| MMSE | Minimum Mean Square Error |
| OFDM | Orthogonal Frequency Division Multiplexing |
| OSI | Open Systems Interconnection |
| PDF | Probability Density Function |
| PSK | Phase-Shift Keying |
| QAM | Quadrature Amplitude Modulation |
| SNR | Signal to Noise Ratio |
| XOR | Exclusive OR (Gate circuit) |
| ZFE | Zero Forcing Equaliser |

# Zusammensammlung

Der in dieser Arbeit präsentierte MS-Dekoder ist die, das Gesamtoptimum empfangende Methode eines Trelliskodierten, modulierten Signals in einer Multipath Umgebung. Er kombiniert eine ausgleichende und eine convolutional Dekodierung in einem einzelnen vereinheitlichten Prozess, der auf dem relativen Trellisdiagram basiert. Vereinfacht beschrieben nutzt es die, durch den Kodierungsprozess eingeführten Abhängigkeiten um eine effizientere Entzerrung im vereinheitlichten Prozess zu ermöglichen. Eine effizientere Entzerrung sorgt für weniger falsche Resultate und macht die Kodierungsaufgabe zugleich einfacher, da die minimal mögliche Information in dem Kodierungsvorgang „verloren“ geht.

Bezüglich der Rechen- und Designkomplexität des eingesetzten MS decoder wurde keine nennenswerte zusätzliche Belastung festegestellt. Wichtig ist, dass jegliche zusätzliche Behinderung nicht von den Kanal-Eigenschaften abhängt. Sie hängt lediglich von den Eigenschaften des convolutional-Kodierers, der MS-kodiert im Empfänger, ab (Referenzkodierer). Man unterscheidet zwei hauptkategorien. Der Einsatz des MS-Dekoders ohne und mit Vergrößerung der Bandbreite. Im ersten Fall übersteigt die Komplexität nicht das $2^{k(C-1)}$-fache der Komplexität des ML-Detektors, wobei $k, C$ die Eingangsbits und die eingeschränkte Länge des eingelegten Referenz-Dekoders sind. In vielen Fällen nimmt die Komplexität eher ab. Im zweiten Fall nimmt die Komplexität fallspezifisch zu oder ab. Jeder Zuwachs überschreitet jedoch knapp die vordefinierte Grenze von $2^{k(C+1)}$ mal der Komplexität des ML-Detektors, wobei $k, C$ wobei $k, C$ die Eingangsbits und die eingeschränkte Länge des eingelegten Referenz-Dekoders sind.

Es gibt viele Möglichkeiten für den Einsatz des MS-Decoders als Standard. Dies hängt immer von den Zielen des Kommunikationssystems ab und davon, was der Hersteller zur Geltung bringen möchte. In einem ausgewogenen Standard, wo Bandbreite, Informations Bit-Rate, Komplexität und Leistung alle wichtig sind, sind enge Relationen zwischen diesen Faktoren festzustellen. Jeder Versuch, einen dieser Faktoren zu verbessern wird einen anderen Faktor verschlechtern, was unerwünscht ist. Eine Änderung der Leistungsanforderungen zum Beispiel führt zu einer Erhöhung der Bandbreite und dem Gegenteil. Der Erfolg von MS Dekoder liegt in der Lockerung dieser engen Abhängigkeiten, so dass mehr Freiheitsgrade für eine gesamte Verbesserung entstehen. Schema 3 and 4 zeigt dies. Wie in Bild 3.4-13 bis Bild 3.4-16 zu sehen ist, geschieht dies ohne zusätzliche Bandbreite und ohne eine geringere informations Bit-Rate wobei wobei eine Leistungsverbesserung erreicht wurde. Der Preis, der dafür bezahlt wird sind $16$ mal mehr Berechnungen und das Speichern der Pfade und Metrik wie im Standard Szenario sowie der Bezug zu ausschließlich einer der ausgeführten Operationen. Es ist eine eher unbedeutende Bedingung, bezüglich der großen Leistungsabnahme die erreicht wurden und bezüglich der Rate, welche heutige Prozessormöglichkeiten überschreiten.

Ein weiterer guter Fall wird in Bild 5.3-2 bis Bild 5.3-5 gezeigt. Man erkennt, dass eine berücksichtigte Leistungsreduzierung auf Kosten einer doppelten Bandbreite erreicht wurde. Dies kann hilfreich für einen besseren Mobilstandard sein. Ohne den Einsatz von MS-Dekoder sind die Resultate für Leistung und Bandbreite schlechter, wie in Bild 5.2-9 bis Bild 5.2-12 zu sehen ist. Zu betonen ist der Schutz von Bits des GSM-Standards, der von hoher Bedeutung ist. Die Resultate der gesamtem bits sind nicht verbessert (Bild 5.3-6 bis Bild 5.3-9), jedoch eine höhere Übersetzungsqualität ist aufgrund der geringeren GMS-Klasse 1 Fehler.

Ein Typischeres Beispiel ist Schema 14. Die relativen Resultate sind in Bild 5.3-11 bis Bild 5.3-14 dargestellt. Wie zu erkennen ist, ist sein Einsatz verglichen mit GSM (Schema 10) höher für geringe Bit-Fehlerraten. Für höhere Bit-Fehlerraten fordert Schema 14 eine höhere Leistung. Bezüglich der Tatsache dass Standards für die Mobile Kommunikation verwendet warden sollen wird Schema 14 von einem Hersteller vorgezogen werden. Der Grund ist, dass die maximale Leistung reduziert wurde. Obwohl mathematisch die mittlere Leistung (Für ale Bit-Fehlerraten) gleich bleibt oder sogar zunimmt, ist die Leistung im Bereich $10^{-4} < BER < 10^{-6}$. wichtig. Die meisten Kommunikationen,

die keinen Signalverlust haben, sind in diesem Bereich. Mit anderen Worten ist die durch die Kurve in Schema 14 abgedeckte die SNR Bandbreite kleiner und mit einem geringeren Maximalwert als die veringerte maximale Abstrahlung.

Es wurde ebenfalls herausgefunden, dass, dass MS-Dekoderen ananwendenden Schemen sowie SchemA 14 durch Differenzierung des PAM Amplitudenlevels weiter verbessert werden könnten. Schema 14 benutzt den 16- ten Grad der Modulation und hat als Resultat, 4- Level PAM Signalisierung. Durch den austausch der traditionellen {±A, ±3A} 4- PAM durch eine {±A, ±2A} Amplitudenübertragung kann eine weitere Verbesserung für die Ausschlag gebenden geringen Bit-Fehlerraten erreicht werden. Soweit die Symbolabbildung betroffen ist, ist des Abbilden mit der von Ungerbock ausgedachten Methode des Geräte-partitionierens empfohlen. Somit nimmt die Distanz zwischen konvergierenden Trellis-Pfaden zu und MS-Dekoder wird weniger Wahrscheinlich verwechseln, welcher der Pfade der Richtige ist. Die Anfälligkeit für Fehler ist somit reduziert.

Der Sachverhalt der Verschachtelung wurde ebenfalls betrachtet. In drahtlosen Anwendungen ist das Verschachteln unabdingbar um durch Mehrwegeschwund oder andere Faktoren herbeigeführte, aufeinander folgende Fehler in der Amplitudenanhäufung zu verhindern. Die Folgerung ist, dass das Verschachteln nicht nach dem Referenzkodierer verwendet werden kann, wenn dieser Kodierer durch einen MS-Decoder in dem Empfänger dekodiert wird. Verschachteln eleminiert alle Abhängigkeiten, die durch den Referenzkodierer eingeleitet wurden. Ohne diese Abhängigkeiten stell MS-Dekoder keine besseren Resultate als ein ML Detektor bereit. Natürlich können das Verschachteln oder andere Operationen vor dem Referenzkodierer im Transmitter eingesetzt werden.

Zusamenfassend wurde erwähnt, dass MS Dekodierung nicht nur in der mobilen Kommunikation, sondern auch in anderen drahtlosen Anwendungen eingesetzt werden kann. Insbesondere kann es dann benutzt warden, wenn ein convolutional Kodierer eingesetzt wird. Wenn sich dieser Kodierer genau vor dem Moduator befindet, können die Resultate entscheidend verbessert warden. Wenn nicht, dann funktioniert MS-Decoder genauso wie ein MLSE bei typischerweise den gleichen Resultaten. Eine gute Möglichkeit zur Nutzung von MS-Dekoder ist, ihn zur Reduzierung von Leistungsanforderungen zu verwenden. Satelliten Kommunikation ist ein gutes Beispiel. Ohne eine strenge Bandweitenbegrenzung könte ein Satellitenstandard verbessert werden um die Leistungsanforderungen zu reduzieren und die Lebensdauen des Transponders zu erhöhen. Geichwohl können Aufgrund der Allgemeingültigkeit von MS decoder verschiedene Anwendungen erfunden werden und verschiedene Standards können verändert werden. In dieser Arbeit liegt der Fokus auf dem GSM 3GPP Standard. Die Sprache der Arbeit ist britisches Englisch.

# Abstract

The project introduces a different approach concerning the equalising and decoding operations when receiving a signal in today's communication systems. It was suggested that these two processes could be combined in order to achieve better decoding results. The key principle is the fact that we perhaps do not need to pay a large price in order to detect a transmission with the optimum ML criterion. The scope is to take advantage of a potential limitation on the number of possible transmitted data sequences so as to reduce computational burden required for equalisation.

Hence, not only can we achieve faster reception but also better, since limiting transmission "variety" is possible to lead in fewer errors. This limitation is accomplished through the use of convolutional coding that is used widely in a lot of applications nowadays. Simulation results were obtained by implementation in programming language C and an MS Visual C++ Compiler. Reduction of bit error rate probability seems satisfactory at this stage, however further work needs to be done as far as consideration for mobile communication is concerned.

# 1. INTRODUCTION

## 1.1 Historical review

A digital communication system is a set of technically compatible electrical and electronic equipment, which is used for providing information transfer between a transmitter and a receiver. The simplest communication system is shown in Fig. 1.1-1. It consists of a sender, a receiver and a medium. The substance of the medium through which data is transferred, varies. Examples are space, telephone wire, optic fibre etc. Messages are being transmitted with the use of appropriate signals, usually of the type:

- Electrical voltages (e.g. co-axial cable, wire),
- Electromagnetic waves (e.g. satellite communications, wireless LAN),
- Light pulses (e.g. optic fibre).

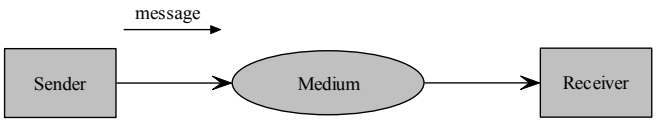


**FIGURE 1.1-1**
A simple communication system.

It can be stated, that the first digital communication system was the electric telegraph, developed by Samuel Morse (1791-1872) in 1837. Morse accomplished a discrete representation of the letters of the English alphabet by corresponding a sequence of dots and dashes to each one. This code is today known as *Morse code*. Morse was also the first one to notice an important parameter in digital communications, intersymbol interference (ISI).

The first modern observations that information and sound could be transmitted through the convulsion of a medium are spotted back in the 17th century. In 1667, Robert Hooke (1635-1703) in Britain described how sound could be transmitted by means of a tightly stretched wire. At the time, more and more scholars were noticing that certain animal muscles contracted in electrical convulsions when brought into contact with silver or copper. The first working telegraph machine is dated a century later. In 1974, Frenchman Claude Chappe (1765-1805) constructed his *Tachygraphe*, used to transmit messages between Paris and Lille. It was a visual semaphore apparatus operated by three men who relayed messages along chains of towers on hilltops five to ten miles apart, using a code of ninety six semaphore signals. Four years later, French Minister of War ordered the apparatus of Claude Chappe to be called the "Telegraphe", a term firstly used by Ignace Chappe, brother of Claude, in April 1793. The word is derived from the two Greek words "tele" (τηλε) meaning "far" and "graphein" (γράφειν) meaning "to write". The name was later applied to the electric telegraph.

In 1800, Alessandro Volta (1745-1827), professor of University of Pavia in Italy, announced his invention of the first electrical battery, the *Voltaic Pile***,** while in 1809, Dr. Samuel Thomas von Soemmering (1755-1830), a surgeon, described to the Munich Academy of Sciences a telegraph apparatus in which, at the receiving end, 26 lines (initially each allotted to a letter) terminated in a container of acid. At the sending station a key, which brought a battery into circuit, was connected as required to each of the line wires. The passage of a current caused the acid to decompose chemically and the message was read by observing at which of the terminals the gas bubbles appeared. This device was the first practically usable electrochemical telegraph (Fig. 1.1-2).

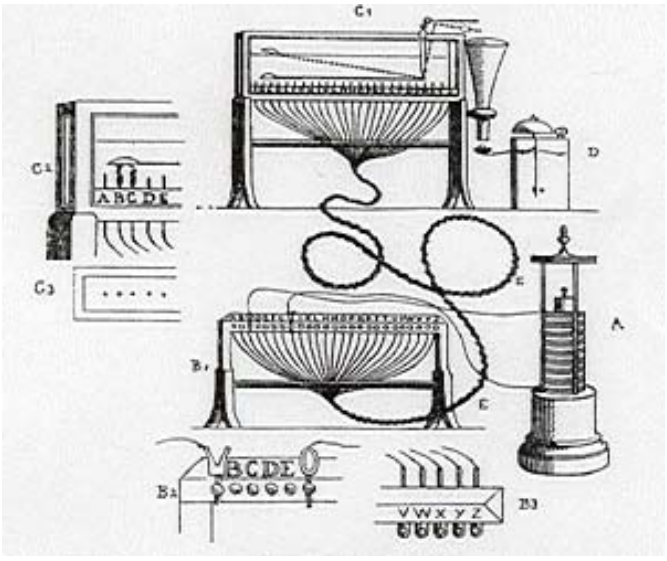


**FIGURE 1.1-2**
First electrochemical telegraph by Soemmering in 1809.

The 19th century meant to be the century of the telegraph as far as telecommunications are concerned. During the first decades, acceptance was not very forthcoming. When Francis Ronalds (1788-1873) offered his recently made copper-wired electrostatic telegraph to British Admiralty in 1816, it was turned down and the existing semaphore system continued to be used. Likewise, noone payed significant attention in 1819 when Hans Christian Oersted (1777-1851), Professor of Physics at Copenhagen University, discovered by chance the link between electricity and magnetism by noticing movements of a magnetic compass when placed near a wire carrying electrical current, or in 1832 when a Russian diplomat in Germany, Baron Pavel Schilling (1780-1836) constructed a revolutionary new telegraph containing a single-needle system and a code to indicate the characters. Global eruption in telegraph usage took place after 1843 when the first telegraph line in America was used, connecting a distance of about 40 miles from Washington to Baltimore. Samuel Morse and his associates put their own made invention into operation while using Morse code for interpretation of the characters. The first message sent was "What hath God wrought".

In the decades to come, the world's first telegraph lines and telegraph stations started to arise in Western Europe and United States. A small but constantly increasing number of people began to use this new means of communication. After the middle of the century public and private telegraph companies were founded. Following the discovery of *Gutta-percha*, inelastic latex and reliable insulator, the first successful submarine telegraph cable was laid across the English Channel between Dover and Cape Gris Nez in 1851. The first successful trans-Atlantic telegraph cable, 1,852 miles in length, was placed between Valentia island of Ireland and Newfoundland in 1866. The first person that employed Gutta-percha to insulate telegraphic cables against moisture was engineer Ernst Werner von Siemens (1816-1892). "Telegraphenbauanstalt Siemens & Halske", the beginning of one of the world's today largest enterprises, was formed in Berlin in 1847 and played an important role in international telegraphy and telephone industry at the time. Inventions like electric dynamo, fascimile telegraphy, telex, and works like Russian telegraphy network and Indo-European telegraph line have been implemented with decisive contribution by Siemens.

On 17 May 1865 in Paris, International Telegraph Union was established by twenty participating countries. The Union was later to become today's International Telecommunications Union (ITU). In 1870, the Post Office public organisation was founded in Great Britain with the absorption of all privately owned inland telegraph systems at a total cost of more than ten million pounds. The Post Office took over a service with 1,058 telegraph offices and 1,874 offices at railway stations using more than 60,000 miles of wire. At the other side of the English Channel, in 1874, Jean Maurice Emile Baudot (1845-1903) invented the *Baudot printing telegraph system* which was the first system to use a

code consisting of five units of equal length. The code she used was a creation of Francis Bacon (1561-1626) implemented back in 1605 for cryptography purposes.

The construction of the first experimental telephone in 1875 by Alexander Graham Bell (1847-1922) marked the beginning of a new age in long-distance communications. Bell was a Scot by birth and immigrated to Brantford, Canada in 1870. In 1872, he moved to Boston in United States and took up an appointment as a teacher of the deaf. Having inherited an interest in the training of deaf children from his father, Bell investigated the artificial reproduction of vowel sounds, resulting in a study of electricity and magnetism and ultimately the development of the telephone.

Within the next years, use of the new device spread rapidly, resulting in formation of telephone companies, exchanges and directories. With the liberalisation of national trunk wires particularly in Britain and US, private call offices were engendered and telephone centres accessed a whole new sector of society, to whom the new technology was largely only a rumour. Long distance telephone calls also started to become reality between adjacent cities. On February the 19th, 1884, Lars Magnus Ericsson of Sweden (1846-1926) combined the transmitter and receiver to produce the earliest telephone handset. Three years later, Oliver Heaviside (1850-1925) propounded the theory that the effect of large electrostatic capacitance of cables could be minimised by increasing their inductance and hence, increasing the distance telephone signals could travel without fading. In 1888, Heinrich Rudolf Hertz of Germany (1857-1894) successfully transmitted electromagnetic radio waves**,** proving that they could be reflected and refracted. The mathematical theory of James Clerk Maxwell (1831-1879) was confirmed. It was the first time radio waves were broadcast and received in laboratory.

At the time, Hertz thought that his discovery was of no significant practical use. When asked by his students at Karlsruhe University he replied: “This is just an experiment that proves Maxwell was right. We just have these mysterious electromagnetic waves that we cannot see with the naked eye. But they are there”. “So, what next?” asked one of his students. Hertz shrugged. He was a modest man, of no pretensions. “Nothing, I guess”. The same year he described in an electrical journal how he was able to trigger electromagnetic waves with his oscillator. A teenager happened to read the article while vacationing in the Alps. Hertz's discovery gave him an idea: why not use the waves set off by Hertz's spark oscillator for signalling? The name of that young man was Guglielmo Marconi (1874-1937). He went back home to Italy to give the idea a try.

Hertz generated waves using an electrical circuit; the circuit contained a metal rod that had a small gap at its midpoint, and when sparks crossed this gap violent oscillations of high frequency were set up in the rod. Hertz proved that these waves were transmitted through the air by detecting them with another similar circuit some distance away. Besides reflection and refraction, he showed that they traveled at the same speed with light but had a much longer wavelength. He also noted that electrical conductors reflect the waves and that they can be focused by concave reflectors. Hertz found that nonconductors allow most of the waves to pass through. These waves, originally called Hertzian waves but now known as radio waves, confirmed Maxwell's prediction on the existence of electromagnetic waves, both in the form of light and radio waves. The English physicist, Oliver Heaviside, said in 1891: “Three years ago, electromagnetic waves were nowhere. Shortly afterward, they are everywhere”.

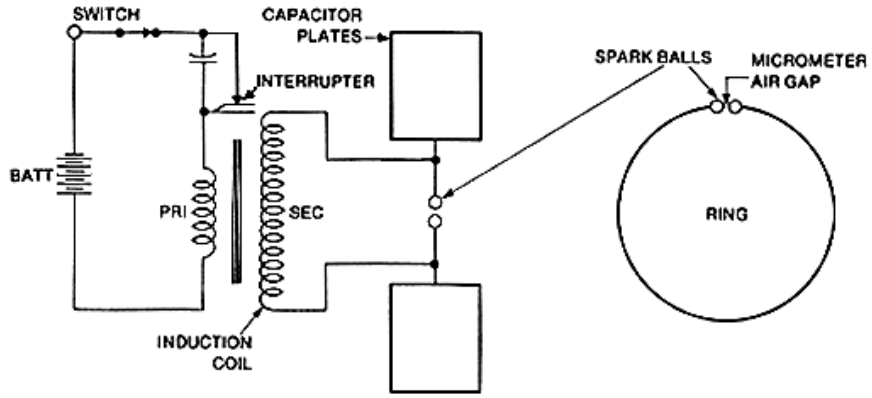


**FIGURE 1.1-3**
First wireless transmitting attempt by Hertz in 1886.

In 1895, Guglielmo Marconi with his mother settled in London from his native Italy. The following year and after a lack of interest from the Italian government, he called upon the Engineer-in-chief of the Post Office to demonstrate his new system of "telegraphy without wires". He gave the first demonstration before members of the Post Office administration on July the 27th. With both transmitter and receiver atop building roofs 300 meters away, signals were satisfactorily recorded. In 1897, *Marconi Wireless Telegraph and Signal Company* was formed and Marconi established the first permanent wireless station at Needles Hotel, Isle of Wight. Earlier in the year he achieved the first ship-to-shore communication, while on a visit to Italy, over a distance of 12 miles. The Italian navy was consequently the first in the world to use radio communication. In 1899, the young Italian bridged the English Channel by radio for the first time when South Foreland, Kent, established communication with Boulogne-sur-Mer by wireless telegraphy. The same year the first maritime distress radio call was made when the East Goodwin Lightship brought the Ramsgate lifeboat to the assistance of the stranded German ship Elbe. Marconi founded more companies in the years to come and built several wireless stations. What is considered a historic day in telecommunications is the 12th of December 1901. It was proved that wireless waves were not devastatingly affected by the curvature of the earth. The first wireless signals were transmitted across the Atlantic from Poldhu Wireless Station in England to Signal Hill, Newfoundland, where Marconi reputedly heard the agreed signal, a succession of three dots, the letter "s" in Morse code. The distance the signal covered was more than 2000 miles.

During the following decades, as "hardware" technology constantly improved telecommunications could not remain unaffected. Many wireless stations were constructed particularly near the coastlines, trunk telephone cables between cities expanded and corporations providing various telephone services were created. 1904 can be considered an important year, when John Ambrose Fleming (1849-1945) invented the *thermionic valve*, a device with two electrodes enabling an electric current to pass through in one direction, but preventing the current from flowing the other way. In addition to its use as a radio wave detector, it was also used as a power supply rectifier, converting alternating current into steady direct current. Fleming's valve is known today as *diode*. In 1907, American Lee de Forest (1873-1961) added a third element to Fleming's thermionic valve to create a *triode*. This had the ability to amplify faint signals, making possible long distance radio and even television communications. The triode was a remarkable invention and was only matched in importance by the invention of transistor which replaced it 41 years later.

Towards the middle of the previous century's second decade, employment of the first amplifiers and permanent repeaters commenced, resulting in prominent reduction of fading when carrying a wired signal for a long distance. Meanwhile, public was becoming more and more acquainted with automatic dialling. The world's first automatic telephone exchange was set up in 1892 in La Porte of Indiana in US. The first one in Europe was constructed and made available to public in 1908 in Hildesheim, Germany. In 1909, the first automatic call center based on a central battery opened in Schwabing of

Munich. Fleetwood of Lancashire in England had the first British automatic exchange for public service in 1922, while in 1923, Munich was the first city with fully automatic local telephone traffic.

During those years, important scientific work was carried out at Bell Telephone Laboratories by a physicist and engineer, Harry Nyquist (1889-1976). Nyquist investigated the problem of determining the maximum signalling rate that can be used over a digital communications channel of a given bandwidth, avoiding or minimising interference [1], [2]. The importance of Nyquist's conclusions still affects modern communication systems nowadays and can be considered in the fourth section of this chapter where one of the most critical parameters in every aspect of digital communication, namely intersymbol interference, is briefly presented.

Despite all the progress that took place in the first half of the 20$^{th}$ century, the big explosion in telecommunications industry did not occur before the work of American Claude Elwood Shannon (1916-2001). Shannon established the mathematical foundations for information transmission and derived the fundamental limits for digital communication systems. He defined channel capacity and proved that errorless transmission through a noisy medium is possible, if the information rate from the source is less than the channel's capacity. That theorem, which was later named as *fundamental theorem of information theory*, came as a shock at the time and gave birth to a new field, information theory.

Up to 1948, communication was thought of as a treatment of electromagnetic waves in order to be transmitted through the air and to be sent down a wire. It was a common belief, that totally errorless transmission was not possible through the use of a noisy medium. This came to collapse after the sensation Shannon's work caused. Shannon first visualised and then proved that errorless transmission is practically realisable with the insertion of extra *redundant* information within the transmitted sequence. After introducing the terms "bit", "redundancy" and "capacity" for the first time, he showed that redundant bits could be used to correct errors in the receiver. In particular, he showed that the noisier the medium is, the more redundancy information is needed and the bigger the transmitted codewords are, inflicting a greater computational burden. Through capacity, he defined the limit of how noisy the medium is allowed to be, or alternatively, how fast propagation can be, in order to achieve errorless transmission. However, he did not devote any of his time to construct such codes accomplishing errorless transmission. With the help of mathematics, he just proved that for every case such a code exists. Practically, Shannon's work is more exploited today and will be more exploited in the future than it was in the middle of the previous century, since computational capabilities were not considerably high at the time. Shannon was also the first engineer to envisage telecommunications in terms of handling binary digits. His publishing in 1948 [3] by Bell Laboratories changed the scientific community's view of telecommunications. In the writer's opinion it constitutes the most important landmark in the history of the field.

In the same year, 1948, Bell Laboratories made another historic publication by announcing the invention of *transistor*. The initial goal of the invention was to counter the excessive waste of energy and the production of exaggerated heat caused by vacuum tubes of triode-amplifiers in telephone networks. The idea behind was to make use of a strange class of materials called semiconductors. With the help of many scientists and engineer John Pierce, who wrote science fiction in his spare time, Bell Labs settled on the name "transistor" combining the ideas of "trans-resistance" with the names of other devices like thermistors.

The beginning of transistor massive production helped telephone devices and cables to be set all over Western Europe and US. Telephony became reality for the majority of citizens. Circuits and lines were of course analogue; quality of service was nevertheless more than satisfactory through the use of efficient amplifiers and repeaters. A first step into digitisation took place in 1964 when trial Pulse Code Modulation (PCM) systems were introduced on junction cables. PCM was not an unknown technique and had been used for the first time between 1943 and 1946 within the cryptographic speech system "Sigsaly", developed by Bell Laboratories. During 70s and 80s, international links as well as long distance fibre cables were gradually becoming digital. Inter-exchange links connecting all major cities and suburbs followed. Today all the regional exchanges and most of the switches are digital.

User advantages include faster and tone dialling and better quality of speech particularly in long distance calls. Advantages for telephone companies are even greater because of the standardisation achieved. In a digital scheme, operator does not have to spare different types of networks for each particular service provided; one network can carry all data relevant to voice, Telex or any other kind of transmission. In addition, no separate network of specially conditioned data lines is needed. Hence, in most countries partial digitisation of the networks of big cities and densely populated areas was carried out relatively fast. Full digitisation came afterwards with the arrival of ISDN in which all the joints are digital and complete end-to-end digital connectivity is provided.

On 6 April 1965, "Intelsat" (Early bird) the first commercial communications satellite, was launched into a synchronous orbit of 22,300 miles. Additionally, the first Internet began by Bolt, Beranek and Newman. Called the "ARPANET" (Advanced Research Projects Agency), it was a network connecting University of California in Los Angeles (UCLA), SRI in Stanford, University of California in Santa Barbara and University of Utah, using 50Kbps circuits. It was completed to its original specification in 1969. It is interesting to mention that Arpanet project was firstly initiated eight years before as an act to increase US competitiveness during the Cold War. Successful launch of Russian satellite "Spoutnik" in 1957 urged US to synchronise research and development projects from several universities. Evolution of Internet took place in the years to come. In March 1972, the first email software was created in order to allow network developers to communicate, while in October 1972 the first mailing list and improved features like listing, selective reading, archiving, replying and forwarding were feasible. In 1992, Internet Society was chartered, triggering the World Wide Web phenomenon.

With the appearance of Internet, digitisation of telephone networks and with hardware companies radically improving micro technology and computational speeds, high speed digital communications became available to the average user. Most of what followed in the last three decades is basically known. Innovations such as DSL, DECT, Wireless LAN, DVB and HTTP protocol contributed in the eruption of telecommunications. In 1992, the first GSM mobile network was put into function by Radiolinja operator in Finland. Technical standardisation and results testing had been carried out for more than seven years under financial assistance of the European Commission. GSM standard was initially launched by Groupe Spéciale Mobile MoU agreement signed by 13 operators and was later defined by European Telecommunications Standards Institute (ETSI) as the internationally accepted digital cellular telephony standard. Exhibiting more than a billion global users today, it constitutes a great technology achievement. Technical details concerning GSM full rate speech channel will be discussed in the fifth chapter, where physical layer[1] modifications are discussed.

[1] The first layer of the Open System Interconnection (OSI) model.

## 1.2 Basic infrastructure

Fig. 1.2-1 illustrates the elements of a typical digital communication system. Input data is first processed by the transmitter. After modulation, data is transferred through the channel and processed by receiver's demodulator, which is usually a matched filter or cross correlator. Then equalisation, deinterleaving, channel decoding, and source decoding take place. More analytically:

**Source encoder**

Message data is being compressed producing a digital signal. In this way less time is required for transmission. Input data can either be a bit sequence, a discrete signal or even analogue information depending on the case. The type of compression depends on the type of data being transmitted as well as the objective of the communication system. For example, in the case of a discrete to discrete transformation, some form of entropy coding may be used (e.g. Huffman), or other kinds such as dictionary coding (e.g. LZW), arithmetic coding, predictive (e.g. BWT) or transform-based coding (e.g. JPEG) etc. In practice, a combination of many types of coding is also used.

As a reference we can use GSM[1] standard. GSM uses an RPE-LPC encoder that performs an analogue to discrete transformation. The analogue input is the voice waveform which is sampled at a rate of 8KHz. Each sample is converted to an 8-bit binary number; therefore a 64Kbps stream is produced and that equals the transmission rate of a regular telephone line. RPE-LPC encoding compresses the information into a 13Kbps synthetic speech bit stream. In this way, a 4:1 data compression ratio is achieved.

The role of source compression is crucial for practical realisation of communication systems. If input data is digital, higher compression ratios can be accomplished. For instance, in processing 24-bit images JPEG standard can result in a 20:1 data compression ratio, without any visible image degradation.

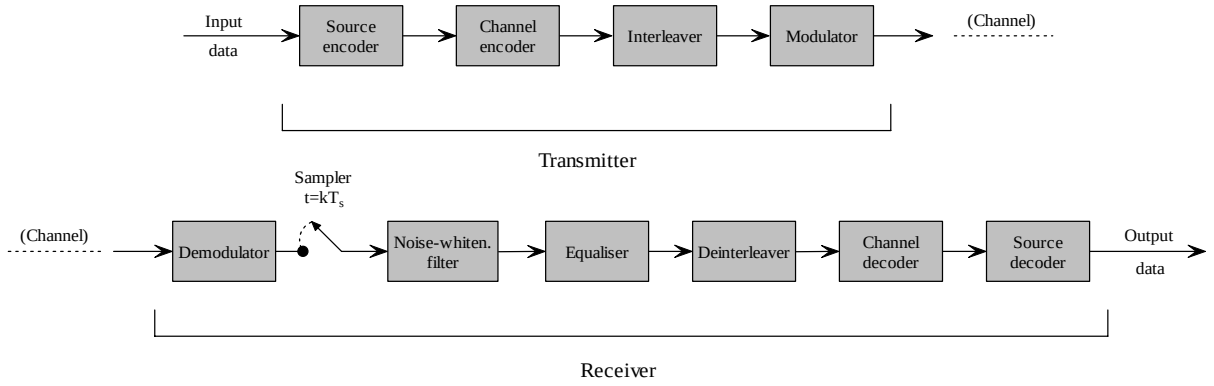


**FIGURE 1.2-1**
A typical digital communication system.

**Channel encoder**

After compression, redundancy bits are added in order to make transmission more resilient against noise and interference. With channel coding, error detection and correction is feasible. The output of channel encoder is a sequence containing more information bits (usually of the order of 50% to 200%) and is passed to the interleaver.

[1] Global System for Mobile Communications.

GSM uses a combination of convolutional encoding and cyclic encoding along with reallocation of bit positions. Each 260-bit burst is at the end converted to a 456-bit burst. Considering the fact that transmission speed is 13Kbps at the input of the channel encoder, we conclude that a 22.8Kbps bit stream is generated at the output.

**Interleaver (II)**

For the formulation of mathematical equations that characterise a channel, as well as for the studying and derivation of bit error rates, it is usually assumed that errors caused by the channel are statistically independent. This is the case when noise is assumed to be AWGN, i.e. an independent Gaussian distribution. In practice however, burst error dependencies are noticed, which result in a number of consecutive errors and which automatically make the AWGN assumption far from true [5]. Multipath fading is a realistic example of such an occurrence.

This phenomenon is countered by interleaving. Data bits are spread in such a way, that the "bursty" channel is transformed into a channel having independent errors. In this way, modelling noise via the AWGN distribution is closer to reality. Additionally, channel coding designed specifically for independent channel errors (i.e. the number of consecutive errors is smaller) can be used. Two types of interleaving are mainly used in practice: Convolutional interleaving and block interleaving. GSM employs a block rectangular interleaver for the control channels and block diagonal interleavers for speech and data channels. In all of simulations presented in the fifth chapter, the block diagonal interleaver of GSM speech channels is used. It is described at that point.

**Modulator**

The modulator firstly *maps* bits into symbols and then converts each symbol into a continuous-time function (waveform), for transmission over a physical channel. The procedure is necessary, because a continuous signal is the only type of signal that can be transmitted over a physical channel in nature. The final shape of the waveform is set by a *transmission* filter. The ideal transmission filter is a square pulse. However, this is not practically possible and instead, trigonometric functions are used; $\sin c$ or the square root of the raised cosine function $rc(t)$

$$rc(t) = \frac{\cos(\pi\beta t/T_s)}{1 - 4\beta^2 t^2/T_s^2} \sin c\left(\frac{\pi t}{T_s}\right) \qquad (1.1)$$

are cases in point. $T_s$ is the symbol interval and $\beta \in [0,1]$ is called the filter *roll-off* factor. The whole modulation filtering process is mathematically described either by basis function $\varphi(t)$, or its equivalent Fourier Transform, $\Phi(f)$. As a simple modulation example, if we assume that transmission is going to be performed with the use of binary Phase-Shift-Keying, modulator firstly corresponds the following waveforms:

$$c\cos\left(\omega_c t + \frac{\pi}{2}\right) = -c\sin(\omega_c t), \qquad \text{to symbols of "-1" ('0' bits)}$$

$$c\cos\left(\omega_c t + \frac{3\pi}{2}\right) = c\sin(\omega_c t), \qquad \text{to symbols of "+1" ('1' bits)}$$

where

$kT_s \le t \le (k+1)T_s$ with $k$ integer,

$c$ is the carrier amplitude,

$\omega_c$ is the angular frequency, equal to $2\pi f_c$,

$f_c$ represents the carrier frequency,

$T_s$ is the symbol duration.

The resulting waveforms are afterwards bandpass filtered and transmitted through a bandlimited channel. A short description of modulation techniques tested in this thesis is carried out in the next section. GSM uses a quaternary modulation scheme, GMSK, as will be later explained.

**Channel**

The physical channel, through which transmission takes place, is described by its impulse response function $h(t)$. In practice, $h(t)$ is time-variant and adaptive techniques are used so that satisfactory estimations can be achieved.

**Demodulator**

In the receiver, the signal is firstly filtered by an analogue filter. The filter's impulse response is most of the times the square root of function (1.1). In other words, transmitter and receiver bandpass filters are identical [6]. Next, demodulation process takes place and the opposite procedure of modulation is performed. The transmitted symbol or bits are extracted through the observation of the received waveforms.

Although optimum demodulation for bandlimited channels is more than just a filter, it is often referred to as *matched filter*. This due to the fact, that the overall demodulator filtering process is matched to the characteristics of the received signal. In fact, the matched filter's impulse response is shaped by the received waveforms. For instance, if we denote with $p(t)$ the convolution of $\varphi(t)$ and $h(t)$, then the matched filter's impulse response will be $p^*(-t)/\|p(t)\|$, where $\|p(t)\|$ is the $l_2$-norm or Euclidean norm of $p(t)$. It is a positive real number and is given by:

$$\|p(t)\|^2 = \int_{-\infty}^{\infty} |p(t)|^2 dt \qquad (1.2)$$

**Noise-whitening filter**

Modulator and demodulator along with the physical channel $h(t)$ constitute the analogue part of the general communication system. After demodulation, sampling takes place and all following processing is again discrete.

However, a significant difficulty appears at this point. At the output of a matched filter, *correlations* appear within the noise sequence. The noise values are not statistically independent [7]. This phenomenon complicates the evaluation of performance of the various equalisation and estimation techniques that are discussed later. As a result, the use of a discrete noise-whitening filter is needed. The filter outputs an uncorrelated sequence or else, a *white noise* sequence. The transfer function of this digital filter is $\|p(t)\| / P^*(z^{-1})$, where:

$P(z)$ is the z-transform of discrete impulse response $p_k$,

$p_k$ is the impulse response produced from sampling $p(t)$, i.e. $p_k = p(kT)$,

where $k = 0,1,2,...$ and $T$ is the sampling period.

$\|p(t)\|$ is difficult to compute in the digital domain through the use of (1.2). Instead, a discrete approximation is used that is given by

$$\|p(t)\|^2 \approx T \sum_{k=-\infty}^{\infty} |p_k|^2 \qquad (1.3)$$

with $T$ and $p_k$ defined as previously. The main advantage of this approach is that it is easily adjustable for a changing $h(t)$, i.e. for different transmitting scenarios. Calculation of $h(t)$ coefficients is not necessary, since differentiation in sampling values $p_k$ provides all necessary information.

**Equaliser**

The goal of an equaliser is to eliminate interference between adjacent transmitted symbols. In most realistic communication scenarios, channel distortion "corrupts" channel response characteristics and is not known to the receiver. This results in the reduction of the noise margin and if left uncompensated, causes high error rates. In simple words, an equaliser tries to cancel the effects of the channel upon the transmitted signal. A linear ZFE equaliser is an example in which this mitigation is apparent. The transfer function of such an equaliser is the reverse of the transfer function of the transmission channel. A linear ZFE equaliser is shown in Fig. 1.2-2.

It is important to note, that stages of noise white-filtering and equalisation can be viewed as an outer equaliser. Processing of the sampled signal through the digital noise-whitening filter can be considered as part of the equalisation process. GSM uses an optimum equalisation technique, called adaptive MLSE or adaptive ML detection, examining all possible symbol sequences and deciding in favour of the *most likely* transmitted [8]. More about equalisers will be apparent in the next chapter.

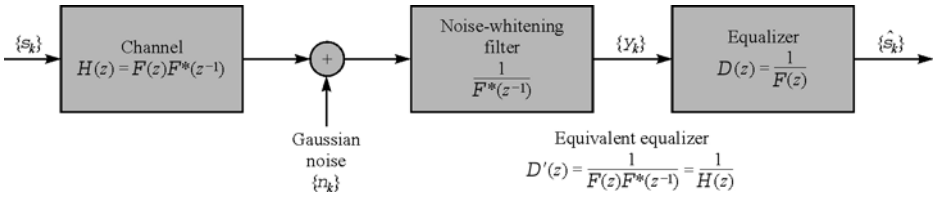


**FIGURE 1.2-2**
Block diagram of channel with a linear zero-forcing equaliser.

**Deinterleaver ($\Pi^{-1}$)**

Deinterleaving involves the distribution of bits of neighbouring bursts in a way reverse of the interleaving procedure. Usually, a few neighbouring bursts participate in each bit "mixing" process.

**Channel decoder**

Channel decoder removes redundancy bits added during the channel encoding procedure. Decoder is aware of the encoding algorithm and hence, "knows" the possible combinations of incoming bit or symbol sequences. The most popular techniques used nowadays for channel decoding are Viterbi algorithm, sequential decoding [9] and MAP decoding. A characteristic decoding example with the use of Viterbi algorithm is presented in section 2.3. A good example of a MAP decoder can be found

in [10]. Most mobile manufacturers employ Viterbi decoding mainly because of its combined optimality and speed.

**Source decoder**

During this last processing of transmitted data, received messages are decompressed using the reverse of the compression algorithm. The procedure could either be a discrete to discrete transformation, or a discrete to analogue transformation depending on the character of the communication system. For instance, in cellular communication, the final data processing that takes place is a discrete to analogue transformation. The analogue output data is afterwards fed into the speakers of the mobile phone.

## 1.3 Digital modulation schemes

Digital modulation is a process in which digital data is transformed into a sinusoidal waveform in order to be transmitted through a band-pass channel such as radio or cable. It can be said that digital modulation is a digital to analogue transformation. Before describing particular techniques, it is important to outline the main signal representation of digital data generated by logic circuits. The representation is called *baseband* PAM signalling, or PCM signalling, or even *baseband modulation*.

### 1.3.1 Baseband signalling

An $M$ - level PAM signal is a sequence of rectangular waveshapes used to represent digital data. Each rectangular waveshape carries information concerning $\log_2 M$ bits. Therefore, 2-level (binary), 4-level (quaternary), 8-level (octernary), 16-level (duo-octernary) and 32-level PAM signals are used. Theoretically there are two kinds of PAM signals, unipolar and bipolar as can be seen in Fig. 1.3-1. In practice, bipolar signalling is preferred. If $A$ is a real number, then the voltage amplitude values of a bipolar PAM signal are usually taken from the set {±A, ±3A, ±5A, ..., ±(M-1)A} depending on the size of $M$. For example, for an octernary PAM signal, $M = 8$ and there is a total of eight different signalling states. Each signalling state takes on a value from the set {-7A, -5A, -3A, -A, +A, +3A, +5A, +7A} and contains three bits. What is important is the ratio of the differences between successive signalling levels and not the absolute values of the differences. In other words, the value of parameter $A$ is not important. What is important are the factors before $A$ which set the amplitude scale between each pair of signalling states. Consequently, for simplicity reasons and without loss of generality, it is often assumed that $A = 1$.

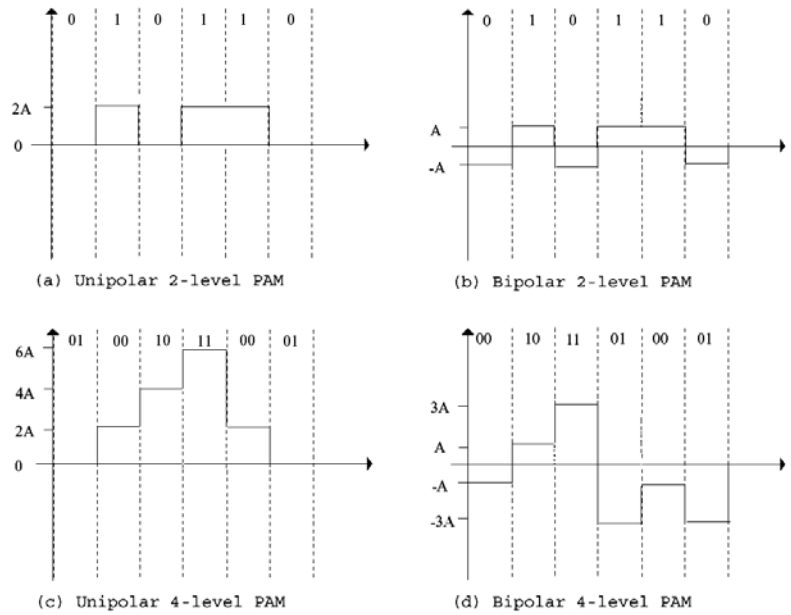


**FIGURE 1.3-1**
Unipolar and bipolar binary and quaternary PAM signals.

The process that corresponds a signalling state to a bit sequence of $\log_2 M$ bits is called *mapping*. There are various mapping techniques depending on the application. One of the most often used is *Gray coding* with which emphasis on the number of bit errors is given. Gray coding makes sure that bit representations of neighbouring symbols differ only in one bit position. In this way, minimisation

of bit errors is pursued since a symbol decision error inside the receiver will likely interpret a symbol as one of its neighbours. An 8-PAM Gray-coded mapping example can be seen below.

| | |
|---|---|
| $"111" \rightarrow 7A$ | $"001" \rightarrow -A$ |
| $"110" \rightarrow 5A$ | $"000" \rightarrow -3A$ |
| $"010" \rightarrow 3A$ | $"100" \rightarrow -5A$ |
| $"011" \rightarrow A$ | $"101" \rightarrow -7A$ |

**FIGURE 1.3-2**
Gray-coded mapping of 8-PAM.

### 1.3.2 Bandlimited modulation

There are two types of digital modulation commonly used in digital transmission systems: Phase-Shift-Keying (PSK) and Quadrature-Amplitude-Modulation (QAM).

$N$ - ary PSK is a phase modulation technique in which digital data is mapped into $N$ discrete phases of a carrier, with $N = 2^n$ and $n \in \aleph^*$. An $N$ - ary PSK signal $s(t)$ may be mathematically expressed as:

$$s(t) = c\cos(\omega_c t + \theta_N) \qquad kT_s \le t \le (k+1)T_s \qquad (1.4)$$

where $c$ is the carrier amplitude, $\omega_c = 2\pi f_c$ is the angular frequency, $f_c$ represents the carrier frequency, $T_s$ denotes the $N$ - PSK symbol duration, $k$ is integer and $\theta_N$ is the carrier phase containing the digital data and having $N$ discrete values given by:

$$\theta_N = \frac{2i-1}{N}\pi \qquad i = 1,\ 2,\ ...,\ N \qquad (1.5)$$

Each carrier phase represents a signalling state and each signalling state contains $\log_2 N$ bits. For instance, in $4$ - ary PSK (or else QPSK), $N = 4$ and the signal has four possible signalling states. Each state contains two information bits.

$N$ - ary QAM is a combined amplitude and phase modulation method with $N \in \{16,64,256,1024\}$. For this reason, it is also referred to as $N$ - ary Amplitude-Phase-Keying (APK). An $N$ - ary QAM signal $s(t)$ can be mathematically represented as:

$$s(t) = \alpha_k \cos\omega_c t - \beta_k \sin\omega_c t \qquad kT_s \le t \le (k+1)T_s \qquad (1.6)$$

where $\omega_c$, $k$, $T_s$ are defined similarly with previously. $\alpha_k$, $\beta_k$ contain the digital data taking values from a $\{\pm A, \pm 3A, ..., \pm(\sqrt{N}-1)A\}$ set. Each pair of $\alpha_k$ and $\beta_k$ represents a signalling element containing $\log_2 N$ information bits. For example, for a 16-ary QAM signal, $N = 16$ and there are 16 possible signalling elements each containing four information bits.

Fig. 1.3-3 shows the block diagram of a QPSK modulator. The input unipolar binary stream of $f_b$ bit rate is converted into two bit streams (inphase and quadrature streams) by a serial to parallel (S/P) converter. Each stream has a bit rate of $f_b/2$. Two unipolar to bipolar (U/B) converters convert these

two streams into two bipolar binary ($\pm 1$) PAM signals. These two binary PAM signals are then modulated by the inphase and quadrature sinusoidal carriers of angular frequency $\omega_c$. This means that the phase of each carrier is shifted by $0^0$ or $180^0$ depending on the amplitude value of the received PAM signal. The two carriers must always have a phase difference of $\pi/2$ as can be seen in the figure. The two modulated signals are added to give a QPSK signal. Finally, the QPSK signal is bandpass filtered at the output of the modulator to remove out-of-band spurious signals caused by the modulation operations. QPSK modems (MOdulator/DEModulator) are extensively employed in operational satellite communication systems.

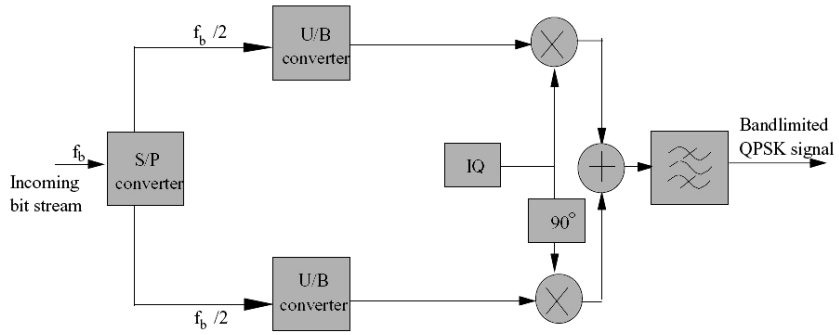


**FIGURE 1.3-3**
QPSK modulator.

It is important to note that after signal reception, the phase of the QPSK signal is used to determine the digital data i.e. the two bits. The phase resulting by modulation of Fig. 1.3-3 takes on one of the four values $\{\pi/4,\ 3\pi/4,\ 5\pi/4,\ 7\pi/4\}$ and that is verified by equation (1.5) for $N = 4$. We can study this with the help of an example. We assume that logic '0' (PAM signal of voltage -1) shifts a carrier by $0^0$ while logic '1' (PAM signal of voltage +1) shifts a carrier by $180^0$. We also assume that $\cos\omega_c t$ and $\cos(\omega_c t + \pi/2)$ waveforms are used for the inphase and quadrature carriers respectively. If information bits '00' are transmitted then both carriers will be shifted by $0^0$ and the generated signal will be

$$\cos\omega_c t + \cos(\omega_c t + \pi/2) = \sqrt{2}\cos(\omega_c t + \pi/4) \qquad kT_s \le t \le (k+1)T_s$$

as expected from (1.4) and (1.5). Likewise, if the information bits are '10' then inphase and quadrature carriers will be shifted by $180^0$ and $0^0$ respectively, producing a QPSK signal

$$\cos(\omega_c t + \pi) + \cos(\omega_c t + \pi/2) = \sqrt{2}\cos(\omega_c t + 3\pi/4) \qquad kT_s \le t \le (k+1)T_s$$

with a different phase, equal to $3\pi/4$. In the same way, '11' and '01' produce the same signal but with different phases $5\pi/4$ and $7\pi/4$ respectively. Hence, the proper combination of inphase and quadrature sinusoidal carriers results in the initially defined PSK signal.

Fig. 1.3-4 illustrates a simplified $N$ - QAM modulator. The input baseband stream of $f_b$ bit rate is converted into two bit streams of $f_b/2$ each. A serial to parallel (S/P) converter is used. The two D/A converters convert the two bit streams into $\sqrt{N}$ - level PAM signals. These signals are amplitude-modulated by inphase and quadrature carriers. The multiplier outputs are then combined and band-pass filtered to give a bandlimited QAM signal.

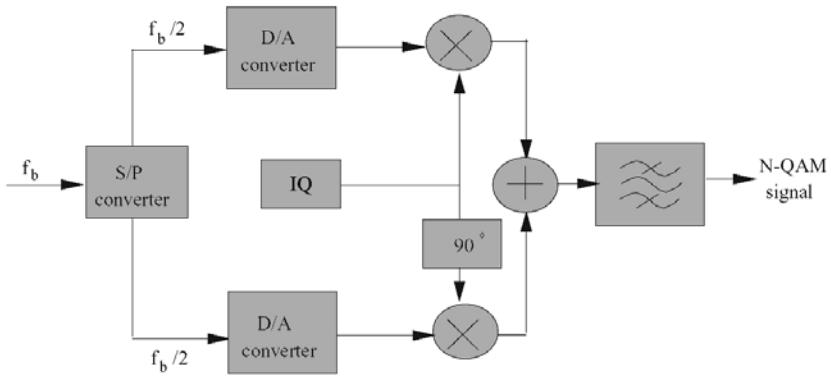


**FIGURE 1.3-4**
$N$ - QAM modulator.

In contrast to PSK, there is no shifting in the individual phases when the inphase and quadrature PAM signals are modulated by the carriers. The amplitude of each PAM signal is simply multiplied with the relevant carrier. Without loss of generality, if we assume that $\cos\omega_c t$ and $\cos(\omega_c t + \pi/2)$ functions are used for the inphase and quadrature carriers respectively and if we denote with $\alpha_k$, $\beta_k$ the amplitude values of the two PAM signals, then the resulting QAM signal $s(t)$ would be

$$s(t) = \alpha_k \cos\omega_c t + \beta_k \cos(\omega_c t + \pi/2) \qquad kT_s \le t \le (k+1)T_s$$

as expected from (1.6). In simple words, two $\sqrt{N}$ - level PAM signals are combined to produce an $N$ - level QAM signal in the transmitter. Correspondingly, an $N$ - level QAM signal is decomposed into two $\sqrt{N}$ - level PAM signals in the receiver. The received PAM signals are then digitised (sampled); the resulting symbols are sorted serially by a (P/S) converter and all following digital processing inside the receiver initiates from this sequence of $\sqrt{N}$ - level PAM symbols. This is important for the needs of this thesis, since this project has to do with digital receiver operations. Equalisers for example, accept as input this sequence of $\sqrt{N}$ - level PAM symbols for further processing.

Fig. 1.3-5 shows candidate modulations for channel bandwidths and bit rates for systems in Europe and North America. For a fixed achievable bit rate, the less bandwidth available, the higher modulation order is used. This is the reason that high order modulations are called *spectrally efficient*. However, efficient bandwidth utilisation does not come for free. Spectrally efficient modulations demand more power than low order schemes such as BPSK and QPSK. On the other hand, low order modulations allow the manufacturer to save on power and consume more bandwidth; therefore they are usually referred to as *power-efficient* techniques.

| | Bit Rate | Channel Bandwidth | | |
|---|---|---|---|---|
| | | 20 MHz | 30 MHz | 40 MHz |
| Europe | 34 Mb/s | 4-PSK | 4-PSK | 4-PSK |
| | 68 Mb/s | 16-QAM | 8-PSK | 4-PSK |
| | 140 Mb/s | 256-QAM | 64-QAM | 16-QAM |
| | 280 Mb/s | ..... | 1024-QAM | 256-QAM |
| North America | 45 Mb/s | ** | ** | ** |
| | 90 Mb/s | 64-PSK | 16-PSK | 8-PSK |
| | 135 Mb/s | 256-QAM | 64-PSK | 16-QAM |
| | 180 Mb/s | 1024-QAM | 256-QAM | 64-QAM |
| | 270 Mb/s | ..... | 1024-QAM | 256-QAM |

** FCC requires a minimum of 78 Mb/s for common carriers

**FIGURE 1.3-5**
Candidate modulations for telecommunication systems.

In most communication scenarios today, BPSK modulation is not used because of its very low spectral efficiency. Power-limited communication systems employ $4^{th}$ order modulation schemes, such as QPSK or GMSK. The latter is widely used in GSM standard. GMSK operates in a way similar to QPSK. A GMSK modulator accepts two binary signals as input producing a GMSK symbol for every two information bits. Equally, a sequence of binary symbols is generated at the demodulator output in the receiver. The digital section of a GSM receiver handles symbols containing one bit of information each.

In practice, the above mentioned modulation techniques are used with a small differentiation. Instead of modulating the value of a bit or of a group of bits on each carrier, the difference between two consecutive bits or two consecutive groups of bits is used. This is called *differential* encoding/decoding and is usually denoted with a "D" before the modulation name, e.g. DQPSK. Finally, an important issue arises concerning digital simulation of telecommunication systems. Considering the fact that modulating and demodulating procedures are analogue, a question of incompatibility appears if we want to test such a system using only a discrete-computation machine such as a computer. The solution to this problem is to construct a mathematical discrete model that incorporates the analogue components and operates in a similar way. In this way, simulations can be performed without the need of employing analogue components in a lab environment. This model is described in the last section of this chapter.

## 1.4 Intersymbol interference

### 1.4.1 The ISI effect

The spreading of symbols such that the energy from one symbol affects the next or the previous ones in such a way that the received signal has a higher probability of being interpreted incorrectly is called ISI.

As mentioned previously, from the very first attempts of digital transmission in the 19$^{th}$ century, it was noticed that the received signals tend to get elongate and interfere to each other. A short pulse was received as a much-smeared version of the same thing. The problem appeared to be related to the properties of the medium used and the distance of signal travel. This phenomenon affects both wire and wireless communication systems even today, mainly because of the high bit rates needed. Additionally, ISI can also be caused from occasional channel distortions and hardware imperfection.

Figures 1.4-1 to 1.4-3 provide a good way of illustrating the problem. We first consider transmission of data sequence 101101 (Fig. 1.4-1). Instead of the square pulses which are impossible to create in practice since infinite bandwidth is needed, the shaped dotted line is used.

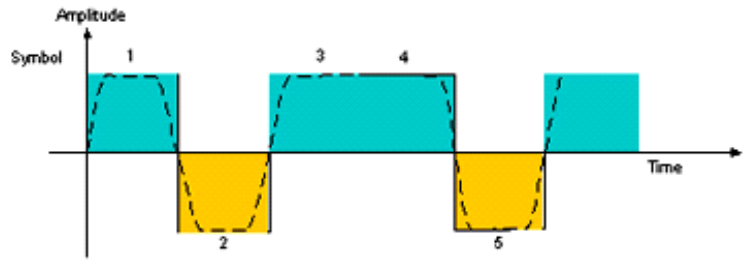


**FIGURE 1.4-1**
Sequence 101101 to be sent.

Fig. 1.4-2 shows each symbol as it is received. It can be seen that the transmission medium creates a tail of energy that lasts much longer than intended. The energy from symbols 1 and 2 goes all the way into symbol 3. Each symbol interferes with one or more of the subsequent symbols. The circled areas show areas of large interference.

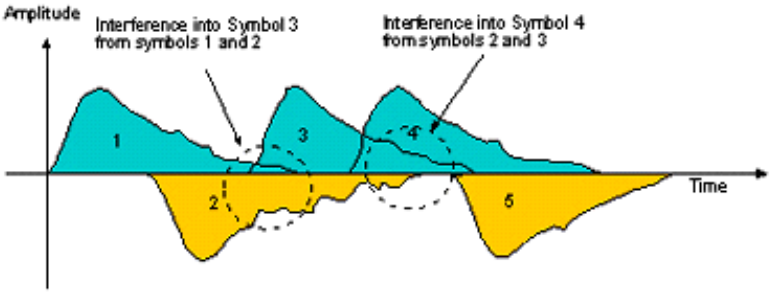


**FIGURE 1.4-2**
Each symbol is spread by the medium.

Fig. 1.4-3 shows the actual signal received as the sum of all these distorted symbols. Compared to the dashed line that was the transmitted signal, the received signal looks much more distorted. The numbered dots show the value of the amplitude at the timing instant. For symbol 3, this value is

approximately half of the transmitted value, which makes this particular symbol more susceptible to noise and incorrect interpretation.

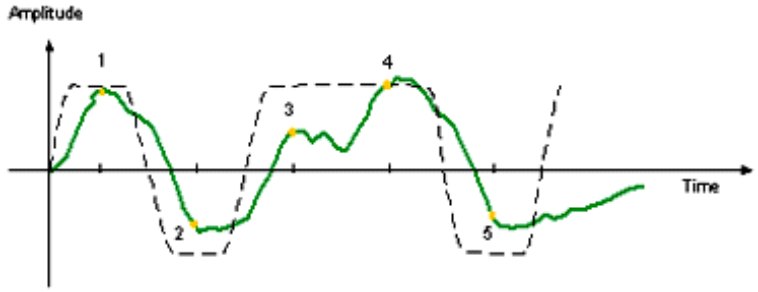


**FIGURE 1.4-3**
Received signal vs. the transmitted signal.

### 1.4.2 Analytical approach

We consider the transmission of a sequence of symbols $s_n$ with the basic waveform $\varphi(t)$. To send the $n$-th symbol $s_n$, we send $s_n\varphi(t-nT)$, where T is the symbol interval and $n = 0,1,2...$ Therefore, the transmitted signal is

$$\sum_n s_n \varphi(t-nT)$$

while the received signal $r(t)$ is given by

$$r(t) = \sum_n s_n\, p(t-nT) + n(t),$$

where $p(t) = (\varphi * h)(t)$ is the channel pulse response (as described in section 1.2),
$n(t)$ is AWGN with power spectral density $N_0/2$.

If a single symbol, e.g. symbol $s_0$ is transmitted, the optimal demodulator is the one that employs a matched filter. The received signal is passed through the matched filter $M(t) = p^*(-t)/\|p(t)\|$ and then the output of the matched filter is sampled at time $t = 0$ to obtain the relevant decision. When a sequence of symbols is transmitted, we can still employ this matched filter to perform demodulation. A reasonable strategy is to sample the matched filter output at time $t = mT$ to obtain the decision for the symbol $s_m$. At $t = mT$ the output of the matched filter $y_m$ is

$$y_m = \sum_n s_n (p * M)(mT - nT) + n_m$$

$$= s_m \|p(t)\| + \sum_{n \neq m} s_n (p * M)(mT - nT) + n_m, \qquad (1.7)$$

where $n_m$ is a zero-mean Gaussian random variable. The first term in (1.7) is the desired signal contribution due to the symbol $s_m$ and the second term contains contributions from the other symbols. These contributions from other symbols constitute the unwanted intersymbol interference.

In the case where $p(t)$ is timelimited i.e. $p(t) = 0$ except for $0 \le t \le T$, it is also $p(t) * p^*(-t) = 0$ except for $-T \le t \le T$. Therefore $(p * M)(mT - nT) = 0, \forall n \ne m$ and there is no ISI. As a result, the demodulation strategy above can be interpreted as matched filtering for each symbol. Unfortunately, a timelimited waveform is never bandlimited. Therefore, for a bandlimited channel, $p(t)$ and hence $p(t) * p^*(-t)$ are not timelimited and hence ISI is in general present. One common way to observe and measure the effect of ISI is to look at the eye diagram of the received signal. The effect of ISI and other noises can be observed on an oscilloscope displaying the output of the matched filter on the vertical input with horizontal sweep rate set at multiples of $1/T$.

For illustration, let us consider that either binary signalling (BPSK) or quaternary signalling (QPSK) is employed. The eye diagrams for these cases are shown in Fig. 1.4-4. The effect of ISI is to cause a reduction in the eye opening by reducing the peak as well as causing ambiguity in the timing information.

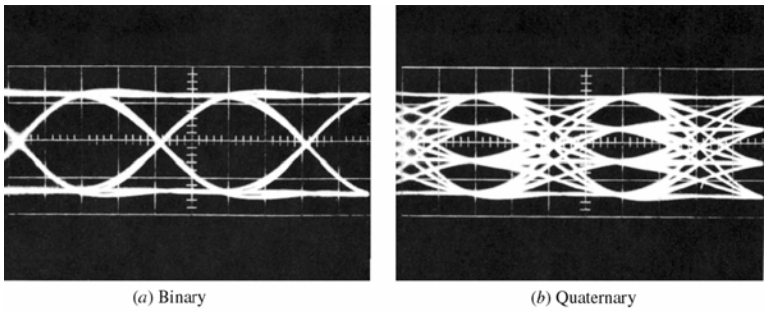


**FIGURE 1.4-4**
Examples of eye patterns for binary and quaternary signalling.

### 1.4.3 Countering ISI

A careful observation on (1.7) reveals that it is possible to have no ISI even if $p(t)$ is not timelimited, i.e., the basic pulse-shaping response $\Phi(f)$ and/or the channel are not bandlimited. More precisely, if $x(t) = p(t) * p^*(-t),$ we can rewrite the decision of $y_m$ in (1.7) as:

$$y_m = s_m \|p(t)\| + \frac{1}{\|p(t)\|} \sum_{n \ne m} s_n x(mT - nT) + n_m \qquad (1.8)$$

Theoretically, there is no ISI if the famous Nyquist condition is satisfied:

$$x(nT) = 0, \forall n \ne 0 \text{ and } x(0) = \alpha, \qquad (1.9)$$

where $\alpha$ is some constant and, without loss of generality, we can set $\alpha = 1.$

This is the time representation of the Nyquist criterion which can also be expressed in the frequency domain. If we set

$$x_\delta(t) = \sum_{n=-\infty}^{\infty} x(nT)\delta(t - nT), \qquad (1.10)$$

[1] Sampling property of the (Dirac) delta function.

then the corresponding FT is

$$X_{\delta}(f)=\frac{1}{T}\sum_{n=-\infty}^{\infty}X\left(f-\frac{n}{T}\right), \tag{1.11}$$

where $X(f)$ is the FT of $x(t)$. The Nyquist condition in (1.9) is equivalent to the condition $x_{\delta}(t)=\delta(t)$ or $X_{\delta}(f)=1$ in the frequency domain. Employing this in (1.11) yields

$$\sum_{n=-\infty}^{\infty}X\left(f-\frac{n}{T}\right)=T, \tag{1.12}$$

which is the equivalent Nyquist condition in frequency domain. It says that the folded spectrum of $x(t)$ has to be flat for not having ISI. In practice however, even if Nyquist condition is satisfied, ISI is in general present and unavoidable. The reasons for this are mentioned in the fifth chapter.

## 1.5 Equivalent discrete model

In dealing with communication channels in the presence of ISI and noise, it is often convenient to represent the analogue part of the system and the following noise-whitening filter with a discrete-time model having a finite number of impulse response coefficients. Modulation, demodulation and noise-whitening filter along with the channel $h(t)$ can be described by means of an FIR filter as shown in Fig. 1.5-1. The filter has coefficients $f_j$ with $-F \le j \le L,$ where $F$ is the number of following interfering symbols and $L$ is the number of preceding interfering symbols.

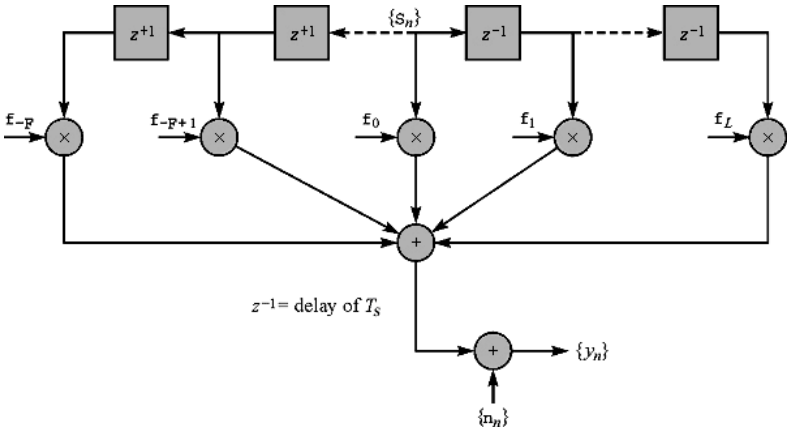


**FIGURE 1.5-1**
Equivalent discrete model of a channel with ISI and AWGN.

Input of the model is the transmitted baseband symbol sequence $s_n$. The output $y_n$ is composed of a linear combination of the input symbols $s_n$ multiplied by corresponding "weights" $f_n$. A noise sample of Gaussian distribution $(n_n)$ is added to the sum. The resulting model is

$$y_n = \sum_{j=-F}^{L} f_j s_{n-j} + n_n \qquad n = 0,1,2,... \tag{1.13}$$

with $s_n = 0, \forall n < 0.$ The output is the convolution of $s_n$ and $f_n$ plus a normally distributed sample. For example, if the total number of preceding and following interfering symbols is four $(F = L = 2)$, then the output at sample instant $t = nT$ is

$$y_n = f_2 s_{n-2} + f_1 s_{n-1} + f_0 s_n + f_{-1} s_{n+1} + f_{-2} s_{n+2} + n_n \tag{1.14}$$

where $n$ is a natural number and $T$ is the symbol period. We will refer to this digital model as *equivalent discrete model*. Additive noise $n_n$ shown in Fig. 1.5-1 is Additive White Gaussian Noise (AWGN). The noise samples at the output of the noise-whitening filter are statistically uncorrelated as mentioned in section 1.2.

Moreover, it is mentioned that when practically realising such a model, the tap coefficients $f_j$ are time-dependent and not fixed. This is caused by time-variations in the physical channel impulse response $h(t)$. Time-variable intersymbol interference effects are therefore produced. In practice, $f_j$ coefficients are real-time estimated through the use of adaptive techniques.

It is important to notice that it does not matter that the model in Fig. 1.5-1 is not causal[1]. This is not a filter that has to be constructed. It is only a mathematical representation of the analogue part of the channel and of the noise-whitening filter. In our next discussions of compensating techniques for ISI and AWGN, the equivalent discrete model will be used.

[1] A system is considered causal when its output depends only on current and previous input values.

# 2. OPTIMUM DECODING & DETECTION

## 2.1 Convolutional codes

### 2.1.1 Basic properties

Convolutional encoding is a multiplexing of two or more different convolutions of the same source data onto a channel. This is accomplished in a continuous manner with the use of shift registers and mod-2 adders. These mod-2 adders are XOR gates. Their inputs are various combinations of the shift register states and their outputs are multiplexed together to form the output stream. The input data stream enters the encoder at a rate of $kb/n$ bits per second, where $b$ is the speed of the output stream, $n$ is the number of mod-2 adders, and $k$ is the number of bits in each register stage. This indicates time redundancy since the input speed is less than that of the output $(k < n)$. Some encoders divide the input data into two separate entities or more, allowing for multiple time redundancy. In practice, encoders are commonly divided like this to match a fixed input rate with a fixed output rate.

Fig. 2.1-1 shows the structure of a typical convolutional encoder. The shift register consists of $C$ stages of $k$ bits each as well as $n$ generators. The input data is shifted into and along the shift register $k$ bits at a time, resulting in an output sequence of $n$ bits. Parameters $C$ and $k/n$ are called *constraint length* and *code rate* respectively. Notation $(C,k,n)$ is usually used in order to refer to a convolutional encoder.

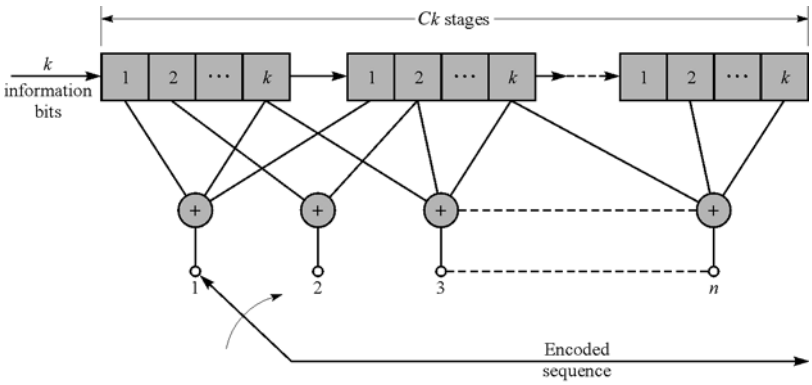


**FIGURE 2.1-1**
Convolutional encoder.

There are three alternative ways to represent the operation of a convolutional code. These are the tree diagram, the Trellis diagram and the state diagram. Examples of these representations are shown in Fig. 2.1-3, 2.1-4 and 2.1-5. The corresponding convolutional code is displayed in Fig. 2.1-2.

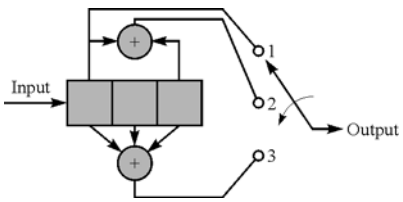


**FIGURE 2.1-2**
$C = 3,\ k = 1,\ n = 3$ convolutional encoder.

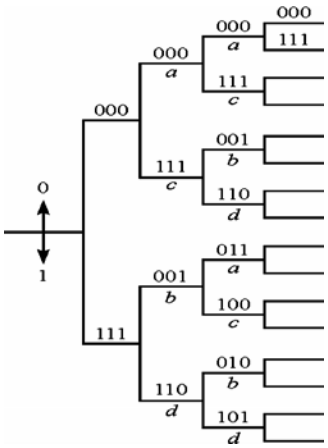


**FIGURE 2.1-3**
Tree diagram for $C = 3,\ k = 1,\ n = 3$ convolutional code of Fig. 2.1-2.

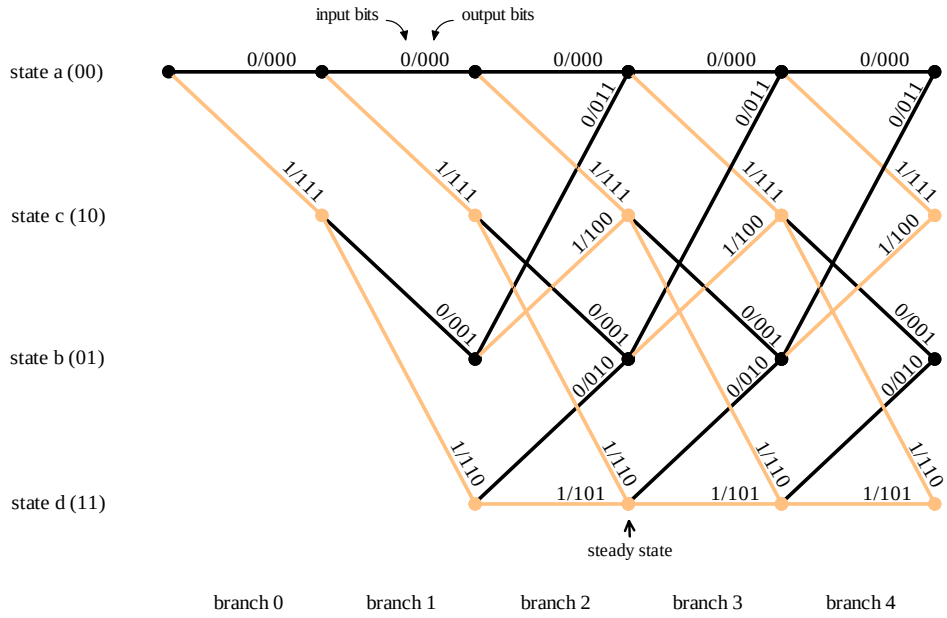


**FIGURE 2.1-4**
Trellis diagram for $C = 3,\ k = 1,\ n = 3$ convolutional code of Fig. 2.1-2.

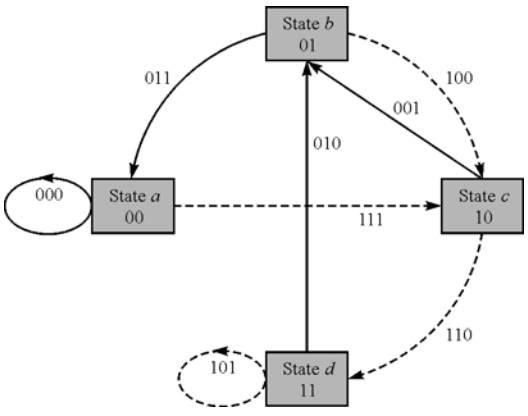


**FIGURE 2.1-5**
State diagram for $C = 3,\ k = 1,\ n = 3$ convolutional code of Fig. 2.1-2.

As can be seen from the state diagram in Fig. 2.1-5, we have to do with a finite-state machine with memory. Each output depends on the input as well as on the current *state*. In other words, convolutional encoding can be described by means of a DFA[1].

The most convenient method to observe the encoder's operation is the Trellis diagram. Trellis diagram makes it easy to view the input and output for many consecutive *transitions*. These transitions are noted in each transition column, usually referred to as *branches* of the Trellis. Each branch contains all possible transitions between two states. The number of total states depends on the number of input bits $k$ and the constraint length $C$ of the encoder. More precisely, it is $2^{k(C-1)}$. It is the total number of bit combinations set by the $k$ input bits and $C$ stages of the shift register. The $k$ bits of the last stage of the register do not contribute to state variation since these bits are shifted out to the right of the register as soon as the next input arrives. It is important to mention that the number of possible outgoing transitions emerging from a state, is $2^k$ and not $2^{k(C-1)}$. In other words, one transition leads only to $2^k$ next states, and not necessarily to the total $2^{k(C-1)}$. Of course, in the special case where $C = 2,$ these numbers are identical.

In particular, in Fig. 2.1-4 there are $2^{k(C-1)} = 4$ total states, each one having $2^k = 2$ possible outgoing transitions. A transition caused by an input bit of '0' is denoted with a black line, while a transition caused by an input bit of '1' is denoted with a beige line. The input bit and the three output bits of each transition are noted across the transition line. The three output bits are generated by encoder as can be seen in Fig. 2.1-2. States 00, 01, 10, 11 correspond to the four different combinations of the first two bits in the 3-bit register. As expected, after the third branch the same Trellis transitions are repeated in each branch (encoder *steady state*). This is due to the fact that, a fixed current state accepting particular input bits, generates a certain transition to a next state producing particular output bits.

One can easily notice that in some cases, the code can be viewed as a non-binary code. For instance, a code with a rate of $2/4$ can be treated alternatively as a quaternary code accepting one quaternary symbol as input and producing two quaternary symbols at the output. This is extremely practical in cases where $M$ - ary baseband modulation is applied at the output symbols, with $M > 2.$ Moreover, it is significant to understand the limitation on the number of possible combinations in the output symbol sequence that is achieved by the convolutional encoder. This limitation serves as "guidance" to the decoding operation carried out in the receiver. In this way, a received symbol sequence is modified and changed into the most probable possible sequence and errors are detected and corrected. Furthermore, this limitation can lead to practically realisable methods with which ISI can be significantly eliminated without the cost of an equaliser's computational burden. MS decoder presented in the third chapter is a case in point.

Finally, it is important to define an important parameter for convolutional codes, *free distance*. If we take a close look on Trellis of Fig. 2.1-4, we will notice that the smallest merging paths consist of three transitions, i.e. expand in three branches. These paths start from the same state and end to the same state. They differ in the input bit of the first transition. The two input bits of the next two transitions are the same. The output bits of these paths are quite different. The smallest number of different output bits of any pair of such merging paths, is called free distance and is denoted with $d_{FD}.$ The bigger it is, the better properties the convolutional encoder has and the easier it is for the corresponding decoder to detect and correct errors [11]. In other words, $d_{FD}$ is the minimum Hamming distance between any pair of such converging paths, irrespectively of the number of transitions these paths contain. For the convolutional encoder of Fig. 2.1-2, $d_{FD}$ is equal to six. If we represented the Trellis output bits with output real numbers, a similar minimum distance could also be defined. An example of such a representation is mapping of Fig. 1.3-2. The minimum distance in this

[1] Deterministic Finite Automaton

case is a positive real number. It is equal to the minimum sum of the squares of the differences between the corresponding output real numbers of any two converging paths. It is in other words equal with a summation of Euclidean distances. It is named *free Euclidean distance* and is denoted with $d_{FED}$.

### 2.1.2 Viterbi decoder

Upon receiving a sequence of symbols, the decoder has the task of finding the original sequence of symbols that served as input to the convolutional encoder and that generated the transmission. The decoder inspects the Trellis diagram and finds the original sequence by performing the inverse procedure, that is decoding. Because of many kinds of interference, the received sequence is likely to be different from the sequence produced by the encoder. The decoder has to make calculations and track down the most probable transmitted symbol sequence.

Viterbi algorithm is an optimum searching algorithm for finding the most probable sequence through the Trellis. The feature of optimality indicates that the decoder takes into account all possible transmitted sequences and chooses the closest. Selection is made with the use of a Maximum Likelihood criterion. The closest sequence is usually called the *shortest path* since it has the shortest distance from the received symbol sequence. The key principle of Viterbi algorithm arises from the fact that the shortest path of transitions that start from branch $k$ and terminate in branch $m$ through an intermediate branch $l$, definitely includes the shortest path of transitions that start from branch $k$ and end in the intermediate branch $l$. In other words, Viterbi algorithm is based on the fact that the ultimate solution of a complex problem comprises of the ultimate solutions of the individual sub-problems. This principle is called Dynamic Programming as first stated by R. E. Bellman in the late '50s [12].

Searching through the Trellis for the closest path is based on computing the distances of all possible paths. Depending on the mode of operation these distances can be of two kinds: Euclidean distances or Hamming distances. In the former case we have to do with a soft-decision Viterbi decoder, while in the latter with a hard-decision Viterbi decoder. The difference is generated by the fact that in hard-decision decoding, the received symbol values are quantised to the closer next or previous level, so that computational complexity is decreased. This however, results in more susceptibility to decision errors.

In the case of hard-decision decoding, the Hamming distance is simply the number of different bits between the received sequence and the *candidate* sequence. In the case of soft-decision decoding, the minimum square distance is used. Things are a little more complicated, as can be seen in the following example.

We will refer to the encoder of Fig. 2.1-2 with the Trellis of Fig. 2.1-4. For simplicity we will assume that BPSK is employed for modulation, i.e. each transmitted symbol is represented by one bit. We will denote the transmitted symbols (or bits) with $s_j$ where index $j = 0,1,2,...$ and indicates the $(j+1)$-th transmitted symbol. $s_j$ is a binary PAM symbol and takes on the value –1 for the '0' bits and +1 for the '1' bits, as described in section 1.3. We define $y_j$ as the received symbol sequence. The elements of $y_j$ are corrupted by channel interference and distortion, in the way modelled in section 1.5. Their value is a real number and not necessarily –1 or +1. Next we will use $iklm..$ to denote the path through the Trellis which corresponds to choosing state $i$ in the first branch, state $k$ in the second branch, state $l$ in the third branch, state $m$ in the fourth branch and so on, where $i,k,l,m \in \{a,b,c,d\}$, i.e. belong to the set of possible states. We will denote as $M_{iklm..}$ the corresponding metric of such a path.

The input to the decoder is

$$y_j = \sqrt{E_c} s_j + n_j, \qquad j = 0,1,2... \tag{2.1}$$

where $n_j$ represents the AWGN,

$\sqrt{E_c}$ is the transmitted signal power for each coded bit.

Since the channel adds white Gaussian noise to the signal, the input to the decoder is statistically described by the pdf[2]:

$$p(y_j / s_j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{{n_j}^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\left(y_j - \sqrt{E_c} s_j\right)^2}{2\sigma^2}} \tag{2.2}$$

where $\sigma^2$ is the variance of the AWGN.

Note that term $\left(y_j - \sqrt{E_c} s_j\right)^2$ represents the Euclidean distance between the actually received value and the value that would have been received in the absence of ISI and noise. Having assumed that BPSK is used, $s_j^2 = 1$ for every possible value of $s_j$. Hence, the decoder metrics which are calculated by (2.2), differ only with respect to the value $\sum y_j s_j$.

In our example, the metrics for the two possible transitions of the first branch are the first to be computed by the decoder:

$$M_a = \sum_{j=1}^{3} y_j s_j^{(a)} \tag{2.3}$$

$$M_c = \sum_{j=1}^{3} y_j s_j^{(c)} \tag{2.4}$$

where $s_j^{(a)} = -1$ and $s_j^{(c)} = 1 \quad \forall j \in \{0,1,2\}$. In other words, $s_j^{(a)}, s_j^{(c)}$ correspond to the transmitted symbols sequences {0, 0, 0}, {1, 1, 1} respectively. After calculating (2.3) and (2.4), the decoder computes the metrics of the four possible transitions of the second branch, which are $M_{aa}, M_{ac}, M_{cb}, M_{cd}$. During the processing of the third branch, eight metrics are calculated, two for each of the four possible final states. These metrics are $M_{aaa}, M_{aac}, M_{acb}, M_{acd}, M_{cba}, M_{cbc}, M_{cdb}, M_{cdd}$. From these eight possible paths, four with smaller metrics are discarded, one for each of the final possible states $a, b, c$ and $d$. In other words, from each of the pairs $\{aaa, cba\}$, $\{aac, cbc\}$, $\{acb, cdb\}$, $\{acd, cdd\}$, the path with the biggest metric is chosen. Viterbi decoder remains with four most probable paths which are used again during the processing of the next branch. These four *surviving* paths are used as basis for the fourth branch computations. During processing of branch four, decoder does not consider all 16 paths. It considers the eight paths that are generated from the four surviving paths of the third branch processing. The procedure repeats during the processing of all following branches required to decode the received symbols. In each branch, Viterbi decoder calculates eight metrics and stores four surviving paths along with their metrics. It is important to note, that not all symbol combinations could have been possibly transmitted and thus, not all symbol combinations are examined by the decoder. For instance, $aba$ is not a possible path through the Trellis of Fig. 2.1-4; the metric of such a path needs not to be computed.

[2] Maximisation of a pdf of a random variable maximises the probability to estimate the random variable precisely.

In this way, soft decision Viterbi decoder calculates the metrics of all candidate paths. After the processing of each branch, $2^{k(C-1)}2^{k}$ metrics are computed and $2^{k(C-1)}$ paths along with their metrics are stored. Final decision on which of the four surviving paths to choose is usually made at the end of each information segment. This is accomplished with the scarce transmission of a group of $k(C-1)$ '0' bits that are known to the decoder and are used to reset it to the first state. More extensive information about convolutional codes and Viterbi decoding can be found in [13], [14].

## 2.2 Equalisers

### 2.2.1 Equalisation techniques

In the first chapter, we introduced the equivalent discrete model describing modulation, transmission through the channel, demodulation and the digital noise-whitening filter. The basic design principles of the demodulator filter for bandlimited transmission were also described. During the analysis in these chapters, we assumed that the receiver knows the channel response characteristics. We assumed that these characteristics (ideal or not) remain the same and can be estimated with absolute accuracy. In practice, the response of the channel is not known with sufficient precision so that optimum modulator and demodulator filters can be designed. For example, in digital communication over a transmission between two telephone terminals, the channel will be different each time a communication is established. There are also other examples of mobile or radio channels that are met in practice, and whose response characteristics are not fixed but time-variant. In such cases, it is impossible to construct optimum demodulation filters a priori, as described in sections 1.2 and 1.4.

This phenomenon results in channel distortion which in its turn results in intersymbol interference that is responsible for high error rates. ISI is caused by a number of other factors as well. Hardware imperfection or multipath fading are cases in point. The solution to these problems is designing a receiver that employs a means for countering or reducing ISI significantly. This ISI-compensating device is what we call equaliser.

There are several categories of equalisers of which the most important are linear equalisers, decision feedback equalisers (DFE), maximum a-posteriori (MAP) equalisers and maximum likelihood sequence estimators (MLSE). Linear equalisation is based on the use of an LTI[1] filter with adjustable tap coefficients. Countering ISI is carried out in a simple way either with the use of ZF criterion or MMSE criterion, but with a cost to efficiency as far as compensating severe interference is concerned. On the other hand, decision feedback equalisers provide a good compromise between capability of mitigating ISI and computational complexity. DFEs exploit previously detected symbols to suppress ISI in the current symbol being detected.

Let us see more information concerning the way each equaliser operates. We consider the equivalent discrete model of Fig. 1.5-1. Symbol sequence $s_n$ is transmitted and symbol sequence $y_n$ is received at the output of the noise-whitening filter. We will denote with $d_n$ the impulse response coefficients of the equaliser filter. Theoretically, it would be preferable that the equaliser had an infinite number of coefficients, as will be seen below. However, this is inevitable in practice and it is assumed that a number of $2T+1$ tap coefficients are used. Hence, $d_n$ is defined for $-T \le n \le T,\ T \in \aleph^*$.

**Linear Zero Forcing Equaliser (ZFE)**

Linear equalisers estimate the $n$ - th transmitted symbol $\hat{s}_n$ as

$$\hat{s}_n = \sum_{j=-T}^{T} d_j y_{n-j} \tag{2.5}$$

where $T$ is a design choice. The serial concatenation of the equivalent discrete model and the equaliser can be represented by their convolution:

$$q_n = \sum_{j=-\infty}^{\infty} d_j f_{n-j} \tag{2.6}$$

[1] Linear Time Invariant filter.

Zero Forcing Equaliser combines the equivalent discrete model and equaliser impulse responses to force to zero all but one tap in the resulting filter. The tap coefficients $d_n$ are chosen to eliminate ISI. Given the discrete model's coefficients, $d_n$ is selected in order to get the desired response

$$q_n = \sum_{j=-\infty}^{\infty} d_j f_{n-j} = \begin{cases} 1 & (n=0) \\ 0 & (n \neq 0) \end{cases} \tag{2.7}$$

Taking the z-transform of equation (2.7), we obtain

$$Q(z) = D(z)F(z) = 1$$

or else

$$D(z) = \frac{1}{F(z)} \tag{2.8}$$

where $D(z), F(z)$ denote the z-transforms of $d_n$ and $f_n$ respectively. The equaliser with transfer function $D(z)$ is simply the inverse filter to discrete model $F(z)$. Such an equaliser is called Zero Forcing Equaliser. Assuming that the digital noise-whitening filter, included inside the equivalent discrete model, has a transfer function of $1/F^*(z^{-1})$, it is interesting to note that the cascade of the noise-whitening filter and the Zero Forcing Equaliser can be considered as an outer equivalent equaliser with transfer function

$$D'(z) = \frac{1}{F(z)F^*(z^{-1})} \tag{2.9}$$

The above can be verified in Fig. 1.2-2.

**Linear Minimum Mean Square Error Equaliser (MMSE)**

MMSE is an equaliser whose performance is superior to ZFE. It utilises the minimum mean square error criterion to adjust the equaliser's tap coefficients. We define error estimation

$$\varepsilon_n = s_n - \hat{s}_n \tag{2.10}$$

with corresponding mean square value $E[\varepsilon_n^2]$. With the help of (2.5) we estimate the function $J$ to be minimised:

$$\begin{aligned} J &= \min_d E[\varepsilon_n^2] = \min_d E[(s_n - \hat{s}_n)^2] \\ &= \min_d E\left[ (s_n - \sum_{j=-T}^{T} d_j y_{n-j})^2 \right] \end{aligned} \tag{2.11}$$

The error is minimised by choosing $d_n$ such that the error sequence is orthogonal to the signal sequence $y_{n-l}$, for $|l| \leq T$. In other words, we try to achieve $E[\varepsilon_n y_{n-l}] = 0, \ |l| \leq T$.

**Decision Feedback Equaliser (DFE)**

Decision Feedback Equalisers consist of two main parts, a feedforward filter and a feedback filter (Fig. 2.2-1). The output of the noise-whitening filter $y_n$ is used as input of the feedforward filter. The feedforward filter is usually a linear equaliser as described previously. Its output is added to previously detected symbols and further used by a symbol detector to provide the final decision $\hat{s}_n$. The addition takes place so that ISI caused by previous symbols is removed. The feedback filter is responsible for providing information about these previously detected symbols, erroneous or not. Recalling the equivalent discrete model of Fig. 1.5-1 it is easy to understand that the feedback filter should contain at least $L$ tap coefficients, i.e. the number of previously interfering symbols.

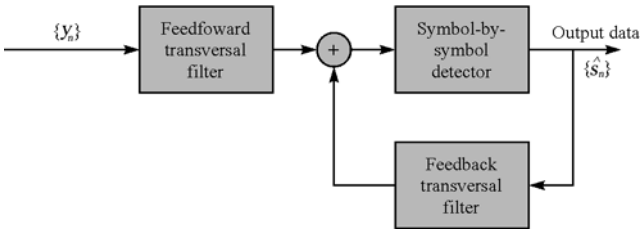


**FIGURE 2.2-1**
A decision feedback equaliser.

Both filters have taps spaced at symbol interval. Assuming that the feedforward and feedback filters have $T_1+1$ and $T_2$ taps respectively and assuming the same notations as previously, the output of a DFE is

$$\hat{s}_n = \underbrace{\sum_{j=-T_1}^{0} d_j y_{n-j}}_{feedforward} + \underbrace{\sum_{j=1}^{T_2} d_j \hat{s}_{n-j}}_{feedback} \qquad (2.12)$$

Because of the factor $\hat{s}_{n-j}$ in the feedback section, i.e. because of the reuse of previously generated outputs, DFE is considered a non-linear equaliser.

**Maximum A-Posteriori Equaliser (MAP)**

MAP equalisers carry out a symbol-by-symbol estimation on the expected values of transmitted sequence symbols $s_n$ based on the a-posteriori probabilities

$$p(\ symbol\ s_n\ is\ transmitted\ \mid\ y\ ) \qquad n=1,2,...,M$$

where $s$ represents the transmitted symbol sequence,
$y$ represents the received symbol sequence,
$M$ is the size of the symbol alphabet, as defined for $M$ - level PAM signals (section 1.3).

MAP detectors make a decision on the transmitted signal in each signal interval based on the observation of vector $y$ in each interval, such that the probability of a correct decision is maximised. Consequently, the probability of error is minimised. During the rest of this thesis we will abbreviate

the aforementioned a-posteriori probabilities as $p(s_n / y)$. Using Bayes' rule, we obtain the expression

$$p(s_n / y) = \frac{p(y / s_n) p(s_n)}{p(y)} \qquad n = 1,2,...,M \tag{2.13}$$

where $p(y / s_n)$ is the conditional pdf function of the observed vector given $s_n$ and $p(s_n)$ is the *a-priori probability* of the $n$ - th transmitted signal (or symbol). At this point we have two cases. The first is when the a-priori probabilities are equal to each other $\forall n = 1,2,...,M$ i.e. when the transmitted symbols are equiprobable. The second is when the symbol probabilities are different because of some a-priori information. A very popular example of the second case is *Turbo equalisation* where feedback of a-priori information is performed [14]. The most popular example of the first case is Maximum Likelihood Sequence Estimator (MLSE) or else, ML detection.

Observing (2.13) we come to interesting conclusions concerning these two cases. These conclusions can be extended for other kinds of processing such as decoding, where similarly, an estimation of the initial bit or symbol sequence is made based on a vector of distorted received bits or symbols. To begin with, denominator $p(y)$ is always the same independently of the choice on $n$. This is reasonable, since there is always one probabilistic choice on received sequence $y$ upon reception. What differs, depending on the choice on $n$, is the nominator of (2.13). MAP detection is equivalent to maximising the product of the conditional pdfs $p(y / s_n)$ with the a-priori probabilities $p(s_n)$. In other words, maximising the a-posteriori probability of a given transmitted symbol sequence is equivalent to maximising the conditional pdf of the received vector (towards this given symbol sequence) multiplied with the initial probabilities of each transmitted symbol included in the examined sequence. If all transmitted symbols are equiprobable, it would be enough to maximise the conditional pdf. This pdf is called the *likelihood function* of the received vector. This is what ML devices perform, such as Viterbi decoder or MLSE described next. They maximise the factor $p(y / s_n)$. Hence, if all transmitted symbols are equally likely to be transmitted, MAP detection yields the same results with ML detection and MAP devices yield the same results as ML devices.

Another distinctive feature firstly introduced by MAP equalisers is the production of *soft output* with the help of Log-Likelihood-Ratio (LLR) modules. In many communication scenarios, it is customary that the equaliser output is used as input of a following decoder. It is preferable to feed the decoder with more information than just bits or symbols. We name this output information as soft output. It comprises of a decision and a probability that the decision is correct. For binary baseband modulation, Log-Likelihood-Ratios shown in (2.14) are a good example.

$$LLR(s_n) = \log \frac{p(s_n = +1 / y)}{p(s_n = -1 / y)} \tag{2.14}$$

The information benefits provided are pretty obvious. If an LLR is a positive real number, then the corresponding bit is a '1'. If an LLR is negative, then we have to do with a '0' bit. Furthermore, the absolute value of the LLR serves as an estimate of the probability that the indicated decision is correct. The bigger it is, the more chances the decision has to be correct. The base of the logarithm does not play a significant role. That is because, for different candidate sequences of transmitted symbols $s_n$ it is not actually decisive to examine the summation of the LLRs of one particular candidate sequence by itself. What is important is the comparison between the values of the LLR summations of two or more different examined sequences, in order to choose the sequence with the greatest sum. Hence, the actual logarithm base does not make any difference in our selection, since it contributes the same to all candidate sequences. The only thing that matters is that the same base is used for every sequence to be examined.

Soft output can be considered for other devices besides MAP equalisers. Soft-input Viterbi decoder is a case in point. Indeed, we recall from the previous section that decisions on an input bit in each branch of the Trellis are made based on the distances of each of the two candidate paths. The bigger distance a path has, the less likely it is to be the correct path. Therefore, we can use the logarithm of the inverse ratio of the two distances in order to generate a soft output, of the type of (2.14). LLRs are extensively employed in communication systems nowadays, particularly with serial or parallel concatenation of convolutional codes. Turbo decoding within receivers of 3G mobile networks is a popular example.

### 2.2.2 MLSE – Viterbi algorithm

Maximum likelihood detection involves the procedure of assigning a metric to each one of the possible transmitted symbol sequences and deciding in favour of the sequence having the larger metric. All possible combinations of transmitted symbol sequences are taken into account. If the symbol alphabet is $M$ - ary and the received symbol sequence consists of $m$ symbols where $m \in \aleph^*$, then the number of possible transmitted symbol sequences is $M^m$.

In the presence of ISI that is caused from the overlapping $L+F$ symbols, ML criterion is equivalent to the problem of estimating the state of a discrete-time finite-state machine. The finite-state machine is the equivalent discrete channel with coefficients $f_k$ and its state at any instant in time is given by the $L$ most recent inputs and the $F$ following inputs, i.e. the state at any time $k$ is

$$State_k = \left(s_{k+F}, s_{k+F-1}, ..., s_{k+1}, s_{k-1}, ..., s_{k-L}\right) \qquad (2.15)$$

where $s_k$ is the transmitted symbol sequence,

$$s_k = 0, \ \forall k \le 0.$$

The channel has $M^{L+F}$ possible states. It is possible to describe the channel by an $M^{L+F}$ - state Trellis diagram and the Viterbi algorithm may be used to determine the most probable path through the Trellis.

The metrics used in the Trellis search are the same with the metrics used in soft-decision decoding of convolutional codes. The procedure begins with the received symbol sequence samples $y_1, y_2, ..., y_{L+1}$, from which the $M^{L+F+1}$ metrics are computed

$$\sum_{k=1}^{L+1} \log p\left(y_k / s_{k+F}, ..., s_{k+1}, s_k, s_{k-1}, ..., s_{k-L}\right)$$

The $M^{L+F+1}$ possible sequences of $s_{L+F+1}, s_{L+F}, ..., s_1$ are subdivided into $M^{L+F}$ groups corresponding to the $M^{L+F}$ states $\left(s_{L+F+1}, s_{L+F}, ..., s_2\right)$. The $M$ sequences in each group (state) differ in $s_1$ and correspond to the paths through the Trellis that merge at a single node. From the $M$ sequences in each of the $M^{L+F}$ states, we select the sequence with the largest probability (with respect to $s_1$). We assign to the surviving sequence the metric

$$M_1\left(S_{L+1}\right) = M_1\left(s_{L+F+1}, s_{L+F}, ..., s_2\right)$$

$$= \max_{s_1} \sum_{k=1}^{L+1} \log p\left(y_k / s_{k+F}, ..., s_{k+1}, s_k, s_{k-1}, ..., s_{k-L}\right)$$

The $M-1$ remaining sequences from each of the $M^{L+F}$ groups are discarded. Thus, we are left with $M^{L+F}$ surviving sequences and their metrics.

Upon reception of $y_{L+2}$, the $M^{L+F}$ surviving sequences are extended by one stage. The corresponding $M^{L+F+1}$ probabilities for the extended sequences are computed using the previous metrics and the new increment which is

$$\log p\left(y_{L+2} / s_{L+F+2}, s_{L+F+1}, ..., s_2\right)$$

Again, the $M^{L+F+1}$ sequences are subdivided into $M^{L+F}$ groups corresponding to the $M^{L+F}$ possible states $(s_{L+F+2}, s_{L+F+1}, ..., s_3)$. The most probable sequence from each group is selected, while the other $M-1$ sequences are discarded.

The procedure continues with the reception of subsequent signal samples. In general, upon reception of $y_{L+k}$, the metrics

$$M_k\left(S_{L+k}\right) = \max_{s_K}\left[\log p\left(y_{L+k} / s_{k+L+F}, ..., s_k\right) + M_{k-1}\left(S_{L+k-1}\right)\right] \qquad (2.16)$$

give the probabilities of the $M^{L+F}$ surviving sequences. Thus, as each signal sample is received, Viterbi algorithm involves first the computation of the $M^{L+F+1}$ probabilities

$$\log p\left(y_{L+k} / s_{k+L+F}, ..., s_k\right) + M_{k-1}\left(S_{L+k-1}\right)$$

corresponding to the $M^{L+F+1}$ sequences that form the continuations of the $M^{L+F}$ surviving sequences from the previous stage of the process. Then the $M^{L+F+1}$ sequences are subdivided into $M^{L+F}$ groups, with each group containing $M$ sequences that terminate in the same set of symbols $s_{L+F+k}, s_{L+F+k-1}, ..., s_{k+1}$ and differ in symbol $s_k$. From each group of $M$ sequences, the one having the largest probability as indicated by (2.16) is selected, while the remaining $M-1$ sequences are discarded. Thus, we are left again with $M^{L+F}$ sequences having the metrics $M_k(S_{L+k})$.

It has been noticed in practice, that these $M^{L+F}$ sequences converge at a point of at least $5L$ branches back in time. The surviving sequences usually originate from the same path and thus a variable symbol detection delay is introduced by truncating the surviving paths to the $D$ most recent symbols, where $D \geq 5L$. It has been proved by simulation that the loss of performance due to this sub-optimum decision is negligible. More about Viterbi algorithm for ML detection including some particular examples and derivation of bit error probability can be found in [15].

## 2.3 Error correction

Viterbi decoding and MLSE presented in this chapter, are two well known cases of optimum decoding and detection respectively. A decoding or equalising process is characterised optimum when it takes advantage of input data, as much as possible. At first, optimality implies that all possible transmitted information sequences must be considered and tested. Next, the criterion through which decisions are made, must be based on the Maximum Likelihood principle [16]. Hamming distance criterion is used to examine different bit or hard symbol sequences [17], [18]. Correspondingly, Euclidean distance criterion[1] is used in order to evaluate "soft" information sequences.

Following, is an example of data transmission through a digital communication system. This example is simple and purely theoretical; it is presented so that the importance of convolutional coding in communication systems nowadays can be viewed in a simple way. Additionally, this example serves as a prerequisite for comprehension of MS decoder operation presented in the next chapter.

In order to give emphasis to the decoding process, no equaliser is used. For simplicity reasons the interleaving operation of reordering the symbols or bits (or block of symbols or bits) is skipped as well. A modulation scheme of BPSK is assumed, along with a convolutional encoder of $R = 2/3$ and $C = 2$ which is shown in Fig. 2.3-1 and Fig. 2.3-2. The amplitude distance between two successive symbols is set to six, i.e. '0' bits are transmitted by –3 while '1' bits are transmitted by +3. This can be accomplished by adjusting the voltage amplitude of the baseband signal accordingly, $A = 3$ (section 1.3).

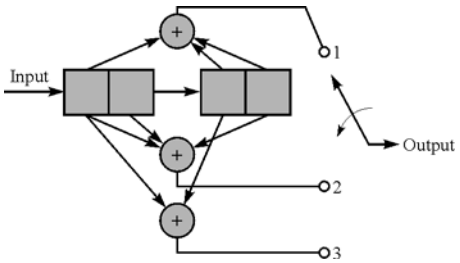


**FIGURE 2.3-1**
$C = 2,\ k = 2,\ n = 3$ convolutional encoder.

[1] Also referred to as "method of least squares".

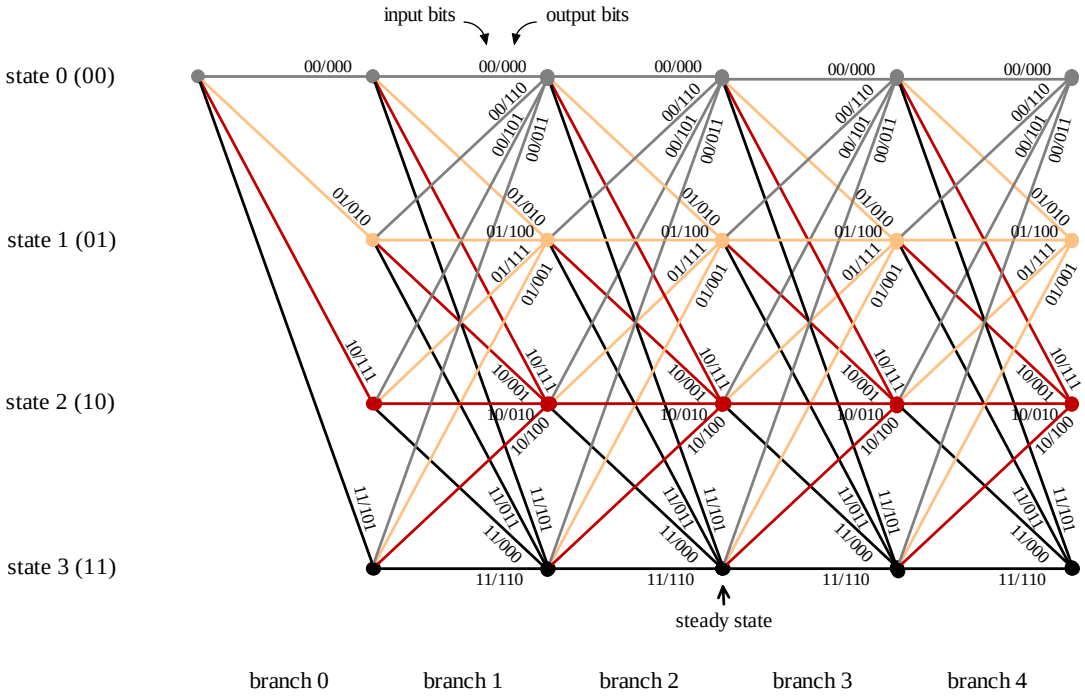


**FIGURE 2.3-2**
Trellis diagram for $C = 2,\ k = 2,\ n = 3$ convolutional code of Fig. 2.3-1.

We consider soft-decision Viterbi decoding and a multipath channel introducing ISI and AWGN. ISI is caused by three preceding symbols, as can be seen from the equivalent discrete model impulse response in Fig. 2.3-3.

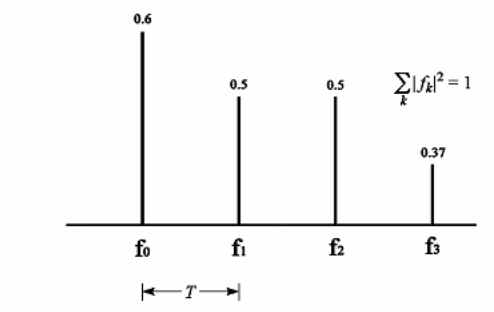


**FIGURE 2.3-3**
Equivalent discrete channel characteristics.

Transmission begins with the extraction of information bits by the source encoder (Fig. 1.2-1). We will examine the processing of a small burst containing only eight bits. The last two bits are '0' and are known to the receiver. The purpose of their use is to flush Viterbi decoder to state '00'. As can be seen from the Trellis in Fig. 2.3-2, the transmission of eight bits corresponds to the first four branches. Two input bits produce three output bits per branch; consequently an output of 12 bits will be transmitted over the channel. Since BPSK is employed, each bit will be mapped to a symbol. The noise values $n_k \quad k = 0,1,...,11$ for these12 symbols are set as

$$n_k = \{0.5,-0.4,-0.1,1.7,2.1,0.8,1,0.2,-0.6,-2.1,-0.9,-0.3\} \qquad (2.17)$$

Assuming that the input bit sequence is

$$\{0,1,1,0,0,1,0,0,\}$$

we are in position to compute the output of the convolutional encoder. 50% redundancy is added and according to Trellis of Fig.2.3-2, the following sequence is produced at the output of the encoder:

$$\{0,1,0,0,0,1,1,1,1,1,1,0\}$$

Following, the bits are mapped into symbols and each symbol is represented by a real value. In our example, one symbol corresponds to one bit and the amplitude distance between two symbols is six. The corresponding sequence of transmitted symbols $s_k$ is

$$s_k = \{-3,3,-3,-3,-3,3,3,3,3,3,3,-3\} \qquad k = 0,1,...,11 \qquad (2.18)$$

After reception and processing through the matched filter and the noise-whitening filter, the received sequence $y_k$ is

$$y_k = \sum_{j=0}^{3} f_j s_{k-j} + n_k \qquad k = 0,1,...,11 \qquad (2.19)$$

with $s_k = 0,\ \forall k < 0.$ Using the $f_k$ values form Fig. 2.3-3 we compute:

$$y_k = \{-1.3,-0.1,-1.9,-1.21,-1.59,-1.51,1.69,3.89,5.31,3.81,5.01,2.01\} \qquad (2.20)$$

Decoder receives the above 'soft' sequence and tries to find the most probable path through branches 0, 1, 2, 3 of the Trellis. We will denote as $ijkl...$ the path through the Trellis that corresponds to the transition of branch 0 leading to state $i,$ to the transition of branch 1 leading to state $j,$ to the transition of branch 2 leading to state $k,$ to the transition of branch 3 leading to state $l,$ and so on. We will use notation $M_{ijkl...}$ for the corresponding metric of this path. Since BPSK is employed, the optimum Viterbi decoder uses the soft decision metrics defined in (2.3) and (2.4). For a given path involving $w$ output symbols $s_k$ the corresponding metric is the sum

$$\sum_{j=0}^{w-1} y_j s_j, \quad \text{where } w \in \aleph^*.$$

$s_j = \pm 3$ depending on the symbols contained in the candidate path. Viterbi decoding procedure begins with the computation of metrics of all four possible paths in the first branch of the Trellis.

**Branch 0**

Each of the four paths of the first branch contains three output symbols $(w = 3)$:

State 0

$$M_0 = -1.3*(-3) - 0.1*(-3) - 1.9*(-3) = 9.9$$

State 1

$$M_1 = -1.3*(-3) - 0.1*(+3) - 1.9*(-3) = 9.3$$

State 2

$$M_2 = -1.3*(+3) - 0.1*(+3) - 1.9*(+3) = -9.9$$

State 3

$$M_3 = -1.3*(+3) - 0.1*(-3) - 1.9*(+3) = -9.3$$

**Branch 1**

During processing of branch 2, the metrics of all 16 possible paths are calculated. Viterbi decoder chooses the best path for each ending state and eliminates the other three. The best path is named *survivor*. It is the one having the larger metric. Note that the metrics $M_0, M_1, M_2, M_3$ of the previous branch are reused.

For the four paths ending in state $s$ where $s \in \{0,1,2,3\}$ the corresponding metrics are

$$M_{is} = M_i + \sum_{j=3}^{5} y_j s_j \qquad i = 0,1,2,3 \qquad (2.21)$$

For example, the path metrics for the first ending state are

State 0

$$M_{00} = M_0 + 3.63 + 4.77 + 4.53 = 22.83 \qquad \text{(survivor)}$$
$$M_{10} = M_1 - 3.63 - 4.77 + 4.53 = 5.43$$
$$M_{20} = M_2 - 3.63 + 4.77 - 4.53 = -13.29$$
$$M_{30} = M_3 + 3.63 - 4.77 - 4.53 = -14.97$$

and $00$ is the surviving path. These paths are different according to reused metrics $M_i$ and to symbol values $s_j$. Decoder does not "know" the elements of sequence (2.18) and can only guess $s_j$ values among $+3$ and $-3$ as specified in Trellis diagram. Similarly, $11$, $12$, and $03$ are the surviving paths leading to each of the rest three ending states. Their metrics are $M_{11} = 14.97$, $M_{12} = 13.17$, $M_{03} = 6.51$.

**Branch 2**

Again, comparisons for each state take place within processing of the third branch. The metrics of the surviving paths of the second branch are reused. All other paths of the previous two branches have already been discarded and are not reconsidered.

For the four paths ending in state $s$ where $s \in \{0,1,2,3\}$ the corresponding metrics are

$$M_{lis} = M_{li} + \sum_{j=6}^{8} y_j s_j \qquad i = 0,1,2,3 \qquad (2.22)$$

where $l \in \{0,1,2,3\}$ and takes on a fixed value dependent on $i$. This value is indicated by the surviving paths of the previous branch. In other words, $l = 0$ when $i = 0$, $l = 1$ when $i = 1$, $l = 1$ when $i = 2$, and $l = 0$ when $i = 3$. The 16 metrics of (2.22) differ to each other according to the value of reused metric $M_{li}$ and to the values of the three symbols $s_j$, $6 \le j \le 8$, that are contained in the

corresponding path. The surviving paths and their metrics for this branch are $M_{030} = 29.04$, $M_{121} = 45.84$, $M_{002} = 55.5$, $M_{113} = 37.5$.

**Branch 3**

In this example's last branch process, decoder "is aware" that the bits at the input of the encoder were '00'. Consequently, the optimum path is chosen among the paths leading to state $s = 0$ (00) while paths ending to other states are ignored:

$$M_{kli0} = M_{kli} + \sum_{j=9}^{11} y_j s_j \qquad i = 0,1,2,3 \tag{2.23}$$

where similarly $k,\ l \in \{0,1,2,3\}$ and take a fixed value dependent on $i$. More analytically:

State 0

$$M_{0300} = M_{030} - 11.43 - 15.03 - 6.03 = -3.45$$
$$M_{1210} = M_{121} + 11.43 + 15.03 - 6.03 = 66.27 \qquad \text{(chosen path)}$$
$$M_{0020} = M_{002} + 11.43 - 15.03 + 6.03 = 57.93$$
$$M_{1130} = M_{113} - 11.43 + 15.03 + 6.03 = 47.13$$

Hence, the most probable path through the Trellis is $1210$ and Viterbi decoder outputs the sequence:

$$\{0,1,1,0,0,1,0,0,\}$$

It is the same as the original input bit sequence. In spite of receiving a much distorted signal like that of (2.20), the decoder achieves in detecting and correcting errors. This is due to the fact that encoding limits the number of combinations of transmitted symbols (or bits). In our short example, $s_1$, $s_5$, and $s_{11}$ are corrupted such that the corresponding received values $y_1$, $y_5$, and $y_{11}$ indicate erroneous transmitted symbols. The decoder, "seeing" that there is no possible output sequence $\{0,0,0,0,0,0,1,1,1,1,1,1\}$ through the Trellis, finds the closest allowable output sequence (ending in state 0). This shift changes information received and eliminates ISI even without the help of an equaliser. Of course, the fact that BPSK was used in the particular example played a significant part as well. The higher the modulation order, the more difficult it is to correct errors for a given SNR.

Decoding procedure operates similarly when involving segments of more than eight bits as an input to the convolutional encoder. Following Viterbi decoder, the output is processed by source decoder and digital data is converted into the type of data that was initially used as input of the communication system.

# 3. MS DECODER

## 3.1 Definition & basic principles

ML detection for a channel with ISI is characterised by a large complexity that increases exponentially with the length of the channel time dispersion. If the size of the symbol alphabet is $M$ and the number of interfering symbols is $L+F$, then Viterbi algorithm requires the computation of $M^{L+F+1}$ metrics and the storing of $M^{L+F}$ sequences along with their metrics, each time a branch is processed through the Trellis. For instance, if we assume $8$- ary baseband modulation and a channel introducing ISI from the tails of 10 symbols, MLSE has to make

$$8^{10+1} \text{ calculations of metrics}$$

and store

$$8^{10} \text{ surviving sequences (paths through the Trellis)}$$
$$8^{10} \text{ metrics, one for each surviving sequence}$$

for each new received symbol. For many channels of practical interest, such a large computational burden is prohibitively expensive to implement.

If we assume that a $(C,k,n)$ convolutional encoder has been used in the transmitter, ML detection and Viterbi decoding procedures of an $M$ - ary baseband PAM signal require

$$Cl_1 = M^{L+F+1} \text{ calculations of metrics} \quad (3.1)$$
$$St_1 = M^{L+F} \text{ surviving sequences (paths through the equalising Trellis)} \quad (3.2)$$
$$St_1 = M^{L+F} \text{ stored metrics, one for each surviving sequence}$$

and

$$Cl_2 = 2^{k(C-1)}2^k \text{ calculations of metrics} \quad (3.3)$$
$$St_2 = 2^{k(C-1)} \text{ surviving sequences (paths through the decoding Trellis)} \quad (3.4)$$
$$St_2 = 2^{k(C-1)} \text{ stored metrics, one for each surviving sequence}$$

where $L+F$ is the number of interfering symbols and $Cl_1,\ Cl_2,\ St_1,\ St_2 \in \aleph^*$.

The key principle of MS decoder is that we do need to go through the storing of all $M^{L+F}$ different combinations of paths in order to achieve optimum equalisation of a Trellis-coded transmission. It is possible to have optimum detection by examining only combinations specified by the convolutional decoder. We can leave equalisation to decoder, having previously adjusted the metrics and operation mode in an appropriate way. The reason why this reduction of complexity can be achieved is the fact that, convolutional coding has already limited the number of possible transmitted symbol sequences. If MLSE is employed at the output of the noise whitening filter, then the MLSE “is not aware” of the useful limitation that has preceded.

We define the *branch* function

$$B: \quad A \longrightarrow Q$$

where $A \subseteq \aleph$ and $Q$ is the set of rational numbers. For an input $a \in A$ the corresponding function value is

$$B(a) = \frac{\log_2 M}{n} a \qquad (3.5)$$

We also define notation $B_a = \lceil B(a) \rceil$ as the upper integer of rational value $B(a)$. If $Cl, St \in \aleph^*$, we will show that with MS decoder, the required complexity for both optimum equalising and decoding can be changed to

$$Cl = 2^{k(C+B_L+B_F)} \quad \text{calculations of metrics} \qquad (3.6)$$

$$St = 2^{k(C+B_L+B_F-1)} \quad \text{surviving paths} \qquad (3.7)$$

$$St = 2^{k(C+B_L+B_F-1)} \quad \text{stored metrics}$$

instead of computations specified by (3.1) to (3.4). With a first look, it can be easily noticed that if $M > 2^k$ a significant reduction of complexity can be achieved.

Similarly with Viterbi decoder, operation of MS decoder is based on the principle of estimating the optimum solution of a complex problem by synthesising the optimum solutions of the individual sub-problems. Decoding procedure is carried out with the help of the typical Trellis diagram of the convolutional encoder. It begins with the computation of all path metrics of the first $C + B_L + B_F$ branches. Each metric corresponds to a different path and each path contains

$$n(C + B_L)/\log_2 M \quad \text{baseband symbols}$$

in the first $C + B_L$ branches and $nB_F / \log_2 M$ baseband symbols in the last $B_F$ branches. In illustrating the Trellis diagram of a convolutional encoder, one can notice that for each transition, the input and output bits are indicated upon the diagram. MS decoder considers the output bits as PAM symbols rather than bits. For instance, for the Trellis shown in Fig. 2.3-2, there are three output bits per transition. Depending on the size of $M$, MS decoder handles the output content of each transition accordingly. E.g. if $\log_2 M = 3$ each branch is considered to correspond to one baseband transmitted symbol. If $\log_2 M = 2$ then each pair of branches is considered to carry three symbols, and so on. The case where $\log_2 M > n$ is a rare occasion which is not often met and is discussed later.

The total number of path metrics calculated in the beginning is given by equation (3.6). These metrics are subdivided into

$$2^{k(C+B_L+B_F-1)} \quad \text{groups,}$$

each group containing $2^k$ metrics. The metrics of each group correspond to a specific path combination of the last $B_L + B_F$ branches included in the first $C + B_L + B_F$ branches of the Trellis. They differ with respect to the transition of the last branch of the first $C$ branches. The above number of groups is easy to verify with simple discrete mathematics. For each of the Trellis states there are $2^k$ possible emanating paths in each branch. In $B_L + B_F$ branches there are $2^{k(B_L+B_F)}$ different combinations of paths for every initial Trellis state. Thus, for the total $2^{k(C-1)}$ Trellis states there are $2^{k(B_L+B_F)}2^{k(C-1)}$ combinations of different paths. Each combination corresponds to a group and each group contains $2^k$ metrics.

MS decoder discards the $2^k - 1$ smaller metrics inside each group. Only one metric and its corresponding path are stored for each group. Therefore, decoder remains with $2^{k(C+B_L+B_F-1)}$ paths and their metrics as indicated by (3.7). This concludes a first stage of the decoder's processing. Practically, the decoder for each combination of transitions that belong to the $(C+1)$- th, $(C+2)$- th, ..., $(C+B_L+B_F)$- th branches has discarded all but one $C$ - th branch transitions.

All this processing is therefore named the $(C+B_L)$- *th branch processing*, in spite of the fact that paths until the $(C+B_L+B_F)$- th branch were considered. No elimination of transitions has yet taken place after the $C$ - th branch.

During the next stage, $(C+B_L+1)$- th branch processing is executed. MS decoder operation extends one branch to the right and performs the same. It computes the path metrics for the first $C+B_L+B_F+1$ branches. The total number of different paths contained in these branches is $2^k$ times greater than the total number of paths of the first $C+B_L+B_F$ branches. However, MS decoder does not calculate or store more metrics than in the previous branch. That is because, in the previous branch processing, transitions were eliminated within the $C$ - th branch. The discarded $C$ - th branch transitions are not considered either at this stage or at any other stage thereafter.

As a result, during the $(C+B_L+1)$- th branch processing, decoder initially receives the $2^{k(C+B_L+B_F-1)}$ surviving paths and their metrics from the previous branch processing. It extends path consideration one branch to the right and computes the $2^{k(C+B_L+B_F-1)}2^k$ metrics of the extended paths. Computing time is saved by using the "old" metrics of the surviving paths in order to calculate the "new" metrics of the extended paths. Each old metric is recalled for the computation of $2^k$ new metrics. This will be later more apparent when the metrics' definition and examples are given.

Like previously, decoder subdivides the new metrics into $2^{k(C+B_L+B_F-1)}$ groups, each group containing $2^k$ metrics. Each group corresponds to a specific path combination of the last $B_L+B_F$ branches of the first $C+B_L+B_F+1$ branches of the Trellis. In other words, each group corresponds to a particular path starting from the $(C+2)$- th branch and ending in the $(C+B_L+B_F+1)$- th branch. Each group's metrics differ with respect to the transition of the $(C+1)$- th branch. There is no path variety for each group concerning the previous branches. No different path selection is possible when "going" backwards in the $C$ - th branch, since any different path choices were eliminated before. No different path selection is possible when going backwards in the first $C-1$ branches either because of the Trellis structure.

The $2^k - 1$ smaller metrics within each group are thrown away. $2^{k(C+B_L+B_F-1)}$ paths and metrics are stored. Consequently, at the end of the $(C+B_L+1)$- th branch processing an equal number of paths and metrics are stored. Transitions from the $C$ - th and the $(C+1)$- th branch have been discarded. The procedure continues for the rest of the branches. If $C \le \Lambda \in \aleph^*$, then at the end of the $(\Lambda+B_L)$- th branch processing transitions from the $C$ - th until the $\Lambda$ - th branch have been discarded. For each of the surviving paths, only one route within these branches has survived.

At the end of a predefined number of branches only one of the surviving paths is finally chosen. As with convolutional decoding, a group of $k(C-1)$ zero bits can be inserted at the end of each burst in order to reset decoder to the first state. Another solution is employment of a *decoding delay*. This is mostly preferred in demanding real-time applications where there is not sufficient time to wait for the end of a long burst. If $t, d \in \Re^+$ with $t \ge d$, at any time $t$ only the most recent decoded information bits are retained in each surviving sequence. These information bits correspond to a time period of $d$

seconds. As each new branch is processed, a final decision is made on the bits received at time $(t-d)$ back in the Trellis, by comparing the metrics of *all* surviving paths between them and deciding in favour of the surviving path having the largest metric. If delay $d$ is chosen sufficiently large, all surviving sequences will likely contain the same input bit in the branch that corresponds back at time $(t-d)$. This does not necessarily mean that all surviving paths will stern from the same node at time $(t-d)$, but it does mean that the different surviving transitions of that branch will likely correspond to the same input bits. Although MS decoder has not been tested in real-time environment, it is expected that a delay of $5D$ branches with $D \geq \max\{C, L\}$ will be sufficient.

It is important to mention that the analysis takes granted that $n$ is equal to or a multiple of $\log_2 M$. If $\log_2 M > n$ or if $n$ is not a multiple of $\log_2 M$, MS decoder's operation becomes complicated and sometimes impractical. Nevertheless, selection of a convolutional encoder with appropriate $n$ would counter any complexity side effects and MS decoder could always be used. For example, for a high $64$ - QAM modulation where $M = 8$, a convolutional encoder with $n$ equal to three or a multiple of three has to be used. An alternative satisfactory approach would be to employ a convolutional encoder such that $L\log_2 M$ and $F\log_2 M$ are a multiple of $n$. That however, is interesting only from theoretical scope, since in practice $L$ and $F$ change over time and no fixed considerations can be made beforehand.

In order to define the metrics, we will use notation $s_k$, $k = 0,1,2,...$ for the transmitted baseband symbol sequence, as in Fig. 1.5-1. For a Trellis path containing $w$ symbols $w \in N^*$, decoder computes the associated metric $M(s_0, s_1, ..., s_{w-1})$. The value of each possibly transmitted symbol $s_k$ is specified by the Trellis diagram. It belongs to a set of $M$ different values, as defined for PAM symbols in section 1.3. The metric of each path is calculated:

$$M(s_0, s_1, ..., s_{w-1}) = -\sum_{k=0}^{w-Ib-1} \left( y_k - \sum_{j=-F}^{L} f_j s_{k-j} \right)^2 \qquad (3.8)$$

where

- $s_k$ is the $(k+1)$ - th transmitted symbol with $s_k = 0 \quad \forall k < 0$,
- $y_k$ is the $(k+1)$ - th received value at the output of the noise-whitening filter, corresponding to the $(k+1)$ - th transmitted symbol
- $f_k$ represents the tap coefficients of the equivalent discrete model describing modulation, transmission through the channel, demodulation (usually with matched filter) and noise-whitening filter,
- $L$ is the number of $f_k$ tap coefficients for $k > 0$,
- $F$ is the number of $f_k$ tap coefficients for $k < 0$,
- $Ib$ is equal to $nB_F / \log_2 M$, the number of symbols in the last $B_F$ branches[1],
- $w$ is the number of symbols that exist in the candidate path, the metric of which MS decoder wants to calculate.

It is assumed that both transmitted symbols sequence $s_k$ and received symbols sequence $y_k$ have samples for $k = 0,1,2...$, i.e. $s_0$, $y_0$ are the first transmitted and received symbols respectively.

[1] $Ib$ is always an integer since $n$ is equal to or a multiple of $\log_2 M$.

Decoder receives values $y_k$ from the output of the noise-whitening filter and tries to make the best possible estimation of the actual symbols transmitted. $f_k$ is estimated through the use of adaptive techniques and algorithms [19], [20]. With the estimation of $f_k$, receiver "finds out" the values of $L$, $F$ as well.

It is apparent from (3.8) that the metrics introduce soft decision operation, since they are nothing else than a summation of "soft" distances between what is received and what should have been received in the case of ML detection. Square distances or else Euclidean distances are used and this indicates that we have to do with ML decoding and detection. MS decoder is thus a soft decision decoder and equaliser, having an operation similar to Viterbi decoding algorithm but with some important modifications.

The first modification is that the metrics used are specified by (3.8). In this way, ML detection takes place and this is accomplished by the term $\sum f_j s_{k-j}$. The second modification has to do with the fact that, unlike Viterbi algorithm, during the processing of a branch in the Trellis, no final decision is made for transmitted symbols existing in the most recent $B_L$ branches, current branch included. This is because all of these most recent branches contain symbols that introduce ISI to symbols of the next branch, which has not yet been processed. Thus, a decision for a "good" or "bad" path containing any of these most recent $B_L$ branches could lead to an optimisation mistake, since a current eliminated "bad" transition may prove to be better than the surviving transition in the future. Finally, a third modification is that, when processing a branch, MS decoder takes into account symbols contained in the following $B_F$ branches. All of these following branches have symbols that interfere to the value of the last symbol existing in the current branch that is being processed. In a way, we have to do with a generalisation of Viterbi decoder for a channel with memory. MS decoder can be considered as a $2^{k(B_L+B_F)}$- dimensional decoder, whereas each dimension constitutes a Viterbi decoder operating with different metrics.

Considering the fact that $2^k$ transitions emanate from each node of the Trellis, we conclude that the number of possible different combinations of paths (with respect to the next $B_F$ branches) is $2^{kB_F}$ for each of the total $2^{k(C-1)}$ nodes. Similarly, the number of possible different combinations of paths with respect to the most recent $B_L$ branches is $2^{kB_L}$ for each of the total $2^{k(C-1)}$ nodes. Hence, for all $2^{k(C-1)}$ nodes in a Trellis branch, the number of possible different surviving sequences is:

$$St = 2^{kB_F} 2^{kB_L} 2^{k(C-1)}$$

and (3.7) is produced. Additionally, considering that the number of new metric calculations (when a branch is processed) is $2^k$ times the number of surviving sequences, (3.6) can be derived:

$$Cl = 2^{kB_F} 2^{kB_L} 2^{k(C-1)} 2^k$$

Obviously, the upper integer of $B(F)$ is the number of the following branches by which MS decoder is extended in comparison to Viterbi decoder. Similarly, the upper integer of $B(L)$ is the number of preceding branches in which no final decisions are made, in contrast to Viterbi decoder.

In representing the metric of a path during processing of the $(b+1)$- th branch, that is branch $b$, we will use the alternative notation

$$M_{\Sigma_0\Sigma_1....\Sigma_b/\Sigma_{b+1}\Sigma_{b+2}....\Sigma_{b+Fol}} = M(s_0, s_1, ..., s_{w-1}) \qquad (3.9)$$

where

$\Sigma_j$ is one of the Trellis states, i.e. $\Sigma_j \in \{0,1,2,...,2^{k(C-1)}-1\}$ $j = 0,1,2,...$

$w = n(b+1)/\log_2 M + n \cdot Fol/\log_2 M$, $Fol \in \aleph$.

Note that branch $0$ is the first branch and that has been taken into account. Integer $w$ is the total number of symbols contained in the path. It is in direct correspondence with equation (3.8). $Fol$ is the number of following branches that contain symbols the values of which interfere to the value of the symbols contained in branch $b$. Besides the processing of the last $B_F$ branches of a transmitting burst, $Fol$ is equal to $B_F$. The metric representation in (3.9) is the same notation that was introduced in sections 2.1 and 2.3. The only difference is that in this case, transitions of branches following the processed branch are also noted. These transitions are stated after the forward slash "/". By looking at the Trellis and at this path symbolisation it is easy to understand which path is mentioned. Each subscript $\Sigma_j$ refers to the ending Trellis state of a transition in branch $j$. In the examples analysed in this thesis, notation (3.9) will be employed.

### 3.2 The algorithm

MS decoder operation is based on the corresponding Trellis diagram of the convolutionally encoded signal. Trellis diagram is easy to comprehend since it shows input and output bits for consecutive state transitions. Different path choices are clearly stated. In this section, the decoding algorithm is described.

As previously, we assume $M$ - ary PAM modulation at the output of a $(C,k,n)$ convolutional encoder. Equivalent discrete model introduces ISI from $F$ following and $L$ preceding symbols. We firstly define a type of binary-logic numbers $bin$ as the concatenation of binary digits. Each of the $bin$ numbers initially consists of

$$Br = C + B_L + B_F \text{ segments,} \tag{3.10}$$

with each segment containing $k$ bits. $B_L$, $B_F$ were defined in the previous section. As algorithm evolves, the size of the $bin$ numbers increases. A segment of $k$ bits is added to the right, each time a new branch is being processed, with the exception of the last $B_F$ branches. If we denote with $\beta_j$, $j = 0,1,2,...$ each comprising bit and with natural number $b$ the branch being processed, then except for the processing of the last $B_F$ branches, $bin$ could be expressed as

$$bin = \underbrace{\beta_0\beta_1...\beta_{k-1}}_{t_0}\ \underbrace{\beta_k\beta_{k+1}...\beta_{2k-1}}_{t_1}\ .....\ \underbrace{\beta_{(b+B_F)k}\beta_{(b+B_F)k+1}...\beta_{(b+B_F+1)k-1}}_{t_{b+B_F}} \tag{3.11}$$

where $t_j$ is the decimal representation of each segment. $t_j$ is an integer sequence the elements of which take $2^k$ different values:

$$t_j \in \{0,1,...,2^k - 1\} \qquad j = 0,1,2,....$$

Next, we define an integer sequence $\Sigma_j$. Each element of $\Sigma_j$ represents one of the Trellis states. It belongs to the set

$$\Sigma_j \in \{0,1,...,2^{k(C-1)} - 1\} \qquad j = 0,1,2,....$$

Their calculation is based on the values of $t_j$ and is conducted through the following forward recursion:

$$\Sigma_0 = t_0 \tag{3.12}$$

$$\Sigma_{j+1} = \left(2^k \Sigma_j + t_{j+1}\right) \bmod \left(2^{k(C-1)}\right) \qquad j = 0,1,2,.... \tag{3.13}$$

Each element of $\Sigma_j$ represents one of the Trellis states. State and branch indexing start from $0$, i.e. state $0$ and branch $0$ are the first state and branch respectively. A second binary number is additionally defined

$$tail = \beta_0\beta_1...\beta_{k-1} \tag{3.14}$$

In contrast to $bin$, $tail$ consists of a fixed number of $k$ bits. We proceed in defining three buffers of $2^{k(Br-1)}$ positions each. Each position of the first two buffers is destined to store a type $bin$ number.

Each position of the third buffer is destined to store a real number. The first position of every buffer is position $0$. Finally, we define $u \in \aleph^*$ and counters $c_1, c_2, c_3 \in \aleph$. $u$ is the total number of branches in each burst to be decoded.

### 3.2.1 Description

As indicated in the previous section, $n$ is always equal to or a multiple of $\log_2 M$. Paths are denoted with the use of (3.9). MS decoder follows the below steps:

1. Initialise $b = C + B_L - 1$ and $c_2 = 0$. Initialise a $bin$ number and set all of its elements to zero, i.e. initialise $\beta_j = 0,\ j = 0,1,...,Br \cdot k - 1$:

$$bin = \underbrace{\beta_0 \beta_1 ... \beta_{k-1}}_{t_0}\ \underbrace{\beta_k \beta_{k+1} ... \beta_{2k-1}}_{t_1}\ .....\ \underbrace{\beta_{(Br-1)k} \beta_{(Br-1)k+1} ... \beta_{Br \cdot k-1}}_{t_{Br-1}}$$

2. Set $c_1 = 0$. Based on the elements of the $bin$ number, calculate decimal values $t_j$. Based on $t_j$ values, calculate the elements of $\Sigma_j$ through equations (3.12) and (3.13). Calculate as many elements of $\Sigma_j$ as to compute the metric of the path

$$\Sigma_0, \Sigma_1, ..., \Sigma_b / \Sigma_{b+1}, \Sigma_{b+2}, ..., \Sigma_{b+B_F}$$

through equations (3.8) and (3.9). Store this metric in position $c_2$ of buffer 3 and store a copy of $bin$ in position $c_2$ of buffer 1. Increase the most significant bit of $bin$. Increase $c_1$ by one.

3. Calculate $t_j$ values based on $bin$. Calculate $\Sigma_j$ values based on $t_j$ values. Calculate as many elements of $\Sigma_j$ as to compute the metric of the path

$$\Sigma_0, \Sigma_1, ..., \Sigma_b / \Sigma_{b+1}, \Sigma_{b+2}, ..., \Sigma_{b+B_F}$$

always through equations (3.8) and (3.9). If and only if this metric is bigger than the metric in position $c_2$ of buffer 3, then replace metric in position $c_2$ of buffer 3 by this one and put a copy of $bin$ in position $c_2$ of buffer 1.

4. Increase $c_1$ by one. If $c_1 < 2^k$ then increase the most significant bit of $bin$ and go to step 3. If $c_1 = 2^k$ then increase $c_2$ by one and proceed normally to step 5.

5. If $c_2 < 2^{k(Br-1)}$ then increase the most significant bit of $bin$ and go to step 2. If $c_2 = 2^{k(Br-1)}$ then increase $b$ by one and proceed to step 6.

6. Set $c_2 = 0, c_3 = 0$. Initialise $tail$ by setting all of its digits to zero. To all $bin$ numbers of buffer 1 add $tail$ to the right, as a suffix.

7. Set $c_1 = 0$. Based on the $bin$ number that is stored in position $(2^k c_2 + c_1)$ of buffer 1, calculate $t_j$. Based on $t_j$, calculate $\Sigma_j$. Based on $\Sigma_j$, compute the metric of the path

$$\Sigma_0,\Sigma_1,...,\Sigma_b/\Sigma_{b+1},\Sigma_{b+2},...,\Sigma_{b+B_F}$$

and store it in position $(2^{k(Br-2)}c_3+c_2)$ of buffer 3. Put a copy of the $bin$ number (of position $(2^k c_2+c_1)$ of buffer 1) into position $(2^{k(Br-2)}c_3+c_2)$ of buffer 2. Increase $c_1$ by one and proceed to step 8.

8. Based on the $bin$ number that is stored in position $(2^k c_2+c_1)$ of buffer 1, calculate $t_j$. Based on $t_j$, calculate $\Sigma_j$. Based on $\Sigma_j$, compute the metric of the path

$$\Sigma_0,\Sigma_1,...,\Sigma_b/\Sigma_{b+1},\Sigma_{b+2},...,\Sigma_{b+B_F}$$

If and only if this metric is bigger than the metric in position $(2^{k(Br-2)}c_3+c_2)$ of buffer 3, then replace metric in position $(2^{k(Br-2)}c_3+c_2)$ of buffer 3 by this one and put a copy of $bin$ (of position $(2^k c_2+c_1)$ of buffer 1) into position $(2^{k(Br-2)}c_3+c_2)$ of buffer 2.

9. Increase $c_1$ by one. If $c_1<2^k$ then go to step 8. If $c_1=2^k$ then increase $c_2$ by one and proceed to step 10.

10. If $c_2<2^{k(Br-2)}$ then go to step 7. If $c_2=2^{k(Br-2)}$ then increase $c_3$ by one and proceed to step 11.

11. If $c_3=2^k$ then go to step 12. If $c_3<2^k$ then set $c_2=0$ and increase the most significant bit of $tail$. Replace the last $k$ bits of all $bin$ numbers of buffer 1 by binary number $tail$. Go to step 7.

12. Increase $b$ by one. If $b=u-B_F$ then set $c_2=0$ and go to step 13. If $b<u-B_F$ then replace $bin$ numbers of buffer 1 by $bin$ numbers of buffer 2 and go to step 6.

13. If $b=u$, set $c_1=1$ and go to step 19. If not, then replace all $bin$ numbers of buffer 1 by $bin$ numbers of buffer 2.

14. Set $c_1=0$. Based on the $bin$ number that is stored in position $(2^k c_2+c_1)$ of buffer 1, calculate $t_j$. Based on $t_j$, calculate $\Sigma_j$. Based on $\Sigma_j$, compute the metric of the path

$$\Sigma_0,\Sigma_1,...,\Sigma_b/\Sigma_{b+1},...,\Sigma_{u-1}$$

and store it in position $c_2$ of buffer 3. Put a copy of the $bin$ number (of position $(2^k c_2+c_1)$ of buffer 1) into position $c_2$ of buffer 2. Increase $c_1$ by one and proceed to step 15.

15. Based on the $bin$ number that is stored in position $(2^k c_2+c_1)$ of buffer 1, calculate $t_j$. Based on $t_j$, calculate $\Sigma_j$. Based on $\Sigma_j$, compute the metric of the path

$$\Sigma_0,\Sigma_1,...,\Sigma_b/\Sigma_{b+1},...,\Sigma_{u-1}$$

If and only if this metric is bigger than the metric in position $c_2$ of buffer 3, then replace metric in position $c_2$ of buffer 3 by this one and put a copy of $bin$ (of position $(2^k c_2 + c_1)$ of buffer 1) into position $c_2$ of buffer 2.

16. Increase $c_1$ by one. If $c_1 < 2^k$ then go to step 15. If $c_1 = 2^k$ then increase $c_2$ by one and proceed to step 17.

17. If $c_2 < 2^{k(C+B_L+u-b-2)}$ then go to step 14. If $c_2 = 2^{k(C+B_L+u-b-2)}$ then proceed to step 18.

18. Increase $b$ by one. If $b = u$ then set $c_1 = 1$ and go to step 19. If $b < u$ then set $c_2 = 0$, replace all $bin$ numbers of buffer 1 by $bin$ numbers of buffer 2, and go to step 14.

19. If metric in position $c_1$ of buffer 3 is bigger than the metric in position $0$ of buffer 3, then replace metric in position $0$ of buffer 3 by metric in position $c_1$ of buffer 3 and replace the $bin$ number in position $0$ of buffer 2 by the $bin$ number in position $c_1$ of buffer 2. Proceed to step 20.

20. Increase $c_1$ by one. If $c_1 < 2^{kB_L}$ then go to step 19. If $c_1 \geq 2^{kB_L}$ then the MS decoded output bit sequence is in position $0$ of buffer 2.

As far as computation of the metrics is concerned, equation (3.8) is always employed. It can be observed that each path metric is a summation of Euclidean distances. Each Euclidean distance corresponds to a symbol in the path. We know that each branch of the Trellis contains $n/\log_2 M$ symbols. We also know that the decoding procedure proceeds one branch at a time. Hence, we understand that each time a branch is being processed, all candidate paths extend $n/\log_2 M$ symbols. Their metrics contain $n/\log_2 M$ more Euclidean distances than the metrics of the paths of the previous branch processing. It is therefore more practical and less time consuming to store the metrics of the previous branch's paths and use them in order to calculate the metrics of the corresponding extended paths in the next branch processing. That is the reason of using buffer 3. Each time a branch is being processed, the metrics of the surviving paths are stored in buffer 3. They are used again in the processing of the next branch to calculate the metrics of the "new" extended paths.

The above algorithm assumed decoding on a burst basis, i.e. decoding a predefined number of $u$ branches within consecutive bursts. As explained previously, for some real-time applications decoding delay can also be introduced. Parameter $u$ is then considered infinite and condition $b = u$ is never satisfied. This means that the algorithm will never reach step 19 and will continue for ever or at least until the end of transmission. It will repeat itself in the loop created within steps 14 to 18. In this case, buffers 1 and 2 need only have space for $bin$ numbers of a specific length. If $l \in \aleph$, then the $bin$ numbers' required length would be

$$l = (5D + B_F + 1)k \text{ bits} \qquad D \geq \max\{C, L\} \qquad (3.15)$$

For all $2^{k(Br-1)}$ binary sequences stored in buffer 2, the first $k$ bits will likely be identical. Each time a branch processing ends (end of step 18), the decoder's output will be the sequence of these $k$ bits.

### 3.2.2 Algorithm complexity

For typical reasons we shall refer to the complexity of the above algorithm. In telecommunication devices, estimating the complexity of the operation algorithm is most of the times meaningless. The reason is that the average floating-point number operations required by the algorithm, are steady and

independent of any factor such as the input values or the precise time the algorithm is used. Average algorithm complexity is linearly proportional to the size of the input data processed, as expected for time-limited applications. For example, in our case we know that the required operations for the execution of one branch processing in the above steps depends exclusively on parameters $k, n, C, L, F, M$ that in no case approach infinite. What can take very large values reaching infinite is only parameter $u$ i.e. the number of branches being processed. We are therefore interested in the relation between the total mathematical operations and $u$.

In general, the average complexity $C_{av}$ of an algorithm can be expressed as the sum:

$$C_{av} = \sum_i p(E_i)t(E_i) \tag{3.16}$$

where $E_i$ is a possible event terminating the algorithm,

$p(E_i)$ is the occurrence probability of that event,

$t(E_i)$ is the number of fl.-point operations that the event requires.

In MS decoding algorithm, there is only one event through which the algorithm terminates. It requires a specific number of fl.-point operations per Trellis branch, say $T$. $C_{av}$ is thus equal to $Tu$. $T$ can be calculated in relation to $k, n, C, L, F, M,$ however this is not necessary since none of these parameters can be infinitely large. We come to the conclusion that $C_{av}$ and $u$ are linearly related

$$C_{av} = O(u) \tag{3.17}$$

as is usually the case in telecommunications. In order to effectively compare two different algorithms referring to the same decoding process, we prefer to calculate the average number of comparisons or/and metrics as well as the necessary memory that is demanded to carry out the processing of a specific data segment. For instance, for convolutional decoders we compare the number of metric comparisons and the number of the metrics stored after the processing of one Trellis branch. More concerning this issue is discussed in section 4.3.

### 3.3 An example

We will present a simple telecommunication example in which MS decoder is employed in the receiver. Its decoding is analysed thoroughly. We employ convolutional encoder of Fig. 2.3-1. Its output is modulated by an 8-level PAM signal and transmitted through a multipath channel. Equivalent discrete model impulse response is shown in Fig. 3.3-1. We have to do with a channel introducing severe ISI. That is told from the amplitude of its Fourier Transform (since it approaches to zero), as well as from the relatively high value of interfering coefficients $f_{-1}$ and $f_1$.

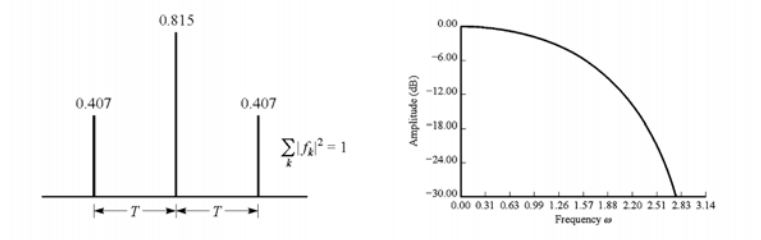


**FIGURE 3.3-1**
Equivalent discrete channel response and frequency characteristics.

We will study transmission of a burst consisting of 16 bits. As usual, the last $k(C-1)$ bits of the burst, i.e. the last two bits are set to zero in order to reset decoder in the receiver. For $k = 0,1,...,7$ we have the AWGN noise values

$$n_k = \{-0.187,-0.039,0.042,0.361,-0.04,0.146,-0.122,0.043\} \qquad (3.18)$$

For simulations performed completely digitally, in other words without the help of analogue components, a sequence such as (3.18) can be generated through mathematical C routines [21]. The noise distribution contains eight samples for this burst, one for each of the 8-PAM symbols modulated at the output of the convolutional encoder. Transmission starts when the first burst

$$\{1,0,1,0,0,1,1,1,0,0,1,1,0,1,0,0\}$$

is fed as input to the convolutional encoder. As can be easily seen from the Trellis of Fig. 2.3-2, the output of the encoder is

$$\{1,1,1,0,1,0,1,1,1,0,1,1,0,1,11,0,1,0,0,1,1,1,0\}$$

Since 8-level PAM is used, each baseband symbol is formed from every three bits. We assume typical Gray-coded mapping of Fig. 1.3-2. The resulting transmitted sequence is

$$\{7,3,7,1,1,-7,-1,5\}$$

After signal reception, matched filter demodulation and processing through the noise-whitening filter, the symbol values $y_k$ are

$$y_k = \sum_{j=-1}^{1} f_j s_{k-j} + n_k \qquad k = 0,1,...,7 \qquad (3.19)$$

with $s_k = 0, \ \forall \ 7 < k < 0.$ Using the $f_k$ values form Fig. 3.3-1 we compute:

$$y_k = \{6.739, 8.104, 7.375, 4.432, -1.667, -5.859, -1.751, 3.711\} \qquad (3.20)$$

The values of (3.20) comprise the input of the following MS decoder, from now on referred to simply as decoder. Its operation begins with processing of the first branch.

Branch 0

Decoder receives the first symbol $y_0$ which corresponds to $s_0$ and starts processing of the first branch. As can be seen from Fig. 2.3-2, each branch contains three output bits and hence corresponds to one 8-PAM symbol. The metrics of the first four paths in branch 0 are computed for every possible transition in the next branch. For each state, there are four possible transitions in branch 1 ($2^k$ transitions), hence the decoder calculates and stores a total of $4 \times 4 = 16$ metrics. At first, the metrics of the four different paths ending in state 0 (after branch 1 transition) are computed:

$$M_{i/0} = -\sum_{k=0}^{0}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 = -\left(y_0 - \left( f_{-1}s_1 + f_0 s_0 \right)\right)^2 \qquad i = 0,1,2,3$$

As always, $s_k = 0, \forall k < 0.$ It is important to note that every value of parameter $i$ corresponds to different $s_0$ and $s_1$ values and hence to a different metric. For instance, the four above metrics are

$$\{M_{0/0}, M_{1/0}, M_{2/0}, M_{3/0}\} = \{-108.26, -5.10, -15.08, -144.89\}$$

Next, the metrics of the paths ending in state 1 (after transition of branch 1) are calculated:

$$M_{i/1} = -\sum_{k=0}^{0}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 = -\left(y_0 - \left( f_{-1}s_1 + f_0 s_0 \right)\right)^2 \qquad i = 0,1,2,3$$

Similarly, these metrics differ with respect to the values of $s_0$ and $s_1$. The estimated values for the second foursome paths are

$$\{M_{0/1}, M_{1/1}, M_{2/1}, M_{3/1}\} = \{-63.41, -40.06, -3.29, -165.15\}$$

The same follows for the other eight paths ending in states 2 and 3 after transition of branch 1. $M_{i/2}$ and $M_{i/3}$, $i = 0,1,2,3$ are similarly calculated and stored.

Branch 1

Decoder receives $y_1$ as the value corresponding to the second transmitted symbol. Having performed processing of branch 0, it starts processing of branch 1. For the first two branches there are 16 total possible paths through the Trellis. However, taking into account the third branch, branch 2, in which there are four possible transitions for each state, we have a total of $16 \times 4 = 64$ different stored paths. Firstly, 16 paths ending in $'.../0'$ are considered, then another 16 paths ending in $'.../1'$ and so on until $'.../3'$.

More analytically, we have four categories of paths, depending on the ending state of the following interfering branch, that is, branch 2. If $s \in \{0,1,2,3\}$ is one of the Trellis states, then for every $s$, the following four path metrics are calculated within processing of first category paths:

$$M_{is/0} = -\sum_{k=0}^{1}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 \qquad i = 0,1,2,3 \tag{3.21}$$

We notice that (3.21) can be expressed as

$$M_{is/0} = M_{i/s} - \left(y_1 - \left( f_{-1}s_2 + f_0 s_1 + f_1 s_0 \right)\right)^2 \tag{3.22}$$

(3.22) shows that decoder can compute (3.21) using $M_{i/s}$, a metric stored after processing of the previous branch. This property saves considerable computing time. Naturally, it is being exploited in calculations of the rest three categories of branch 1 processing, namely $M_{is/1}$, $M_{is/2}$, $M_{is/3}$. The path metrics of these categories are also calculated by (3.21) through different values for $s_0$, $s_1$ and $s_2$. As an example, the second category metrics for $s = 0$ are:

$$\{M_{00/1}, M_{10/1}, M_{20/1}, M_{30/1}\} = \{-219.55, -7.62, -109.93, -224.40\}$$

Branch 2

Up to this point, decoder has completed processing of branches 0 and 1 and has not eliminated any path. However, branch 2 is the $(C + B_L)$ - th branch, as described in section 3.1. Thus, the decoder will start path elimination during processing of this branch. There are $4 \times 4 \times 4 \times 4 = 256$ possible paths through the Trellis for branches 0, 1, 2, 3 and hence a total of 256 metrics are computed. Similarly, all metrics are sorted in four categories, depending on the ending state of branch 3.

We define $s,t \in \{0,1,2,3\}$ as two of the Trellis states. Procedure starts with paths of the first path category, i.e. paths ending in suffix $'.../0'$. For every pair of $\{s,t\}$ four path metrics are calculated:

$$M_{ist/0} = -\sum_{k=0}^{2}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 \qquad i = 0,1,2,3 \tag{3.23}$$

Of these four metrics, decoder stores the largest one and eliminates the other three. The corresponding path of the biggest metric is also stored. For instance, for the first pair $\{s,t\} = \{0,0\}$ of the first path category, the calculated metrics are

$$\{M_{000/0}, M_{100/0}, M_{200/0}, M_{300/0}\} = \{-427.39, -102.44, -356.39, -386.99\}$$

Path $'100/0'$ and its metric are stored. It is called the surviving path or *survivor*. The process continues with all possible $\{s,t\}$ pairs of the rest three categories. The rest three categories include paths ending in suffixes $'.../1'$, $'.../2'$ and $'.../3'$. As always, the metrics differ with each other with respect to the transmitted symbol values $s_k$. The values depend on the path and are specified by the Trellis diagram. There are 16 possible $\{s,t\}$ pairs for each category and four categories in total. Consequently, 256 path metrics are calculated and at the end 64 paths $(16x4)$ and their metrics are stored. The numbers are verified by equations (3.6) and (3.7) as expected.

One can notice that decoder has two options for calculating the metrics of (3.23). Either to calculate them directly from scratch, i.e. using the values $s_0$, $s_1$, $s_2$, $s_3$ taken from the Trellis, or to make recursive use of metrics $M_{is/t}$ stored during processing of branch 1:

$$M_{ist/0} = M_{is/t} - \left(y_2 - \left(f_{-1}s_3 + f_0 s_2 + f_1 s_1\right)\right)^2 \qquad (3.24)$$

In the former case, it can be observed that processing of branch 0 and 1 is not necessary and decoder can start its overall function from branch 2. This is because no path elimination occurred in the previous branches. In general, decoder can start directly from processing of branch $(C + B_L)$, as described in section 3.1. In the example of this section the second option is analysed, so that decoding is more easily understandable. Both cases produce exactly the same output results and make no difference as far as processing of branches after $(C + B_L)$ are concerned.

Branch 3

Upon receiving $y_3$ and having completed processing of branches 0, 1 and 2 decoder repeats a similar procedure with that of branch 2. A total of 256 metrics is calculated and at the end 64 paths and metrics are stored. We denote with $s,t,u \in \{0,1,2,3\}$ three of the Trellis states. For every pair of $\{s,t\}$, four path metrics are calculated within processing of the first category paths:

$$M_{uist/0} = -\sum_{k=0}^{3}\left(y_k - \sum_{j=-1}^{1} f_j s_{k-j}\right)^2 \qquad i = 0,1,2,3 \qquad (3.25)$$

Only the largest of these four metrics survives while the other three are discarded. For each metric there is only one possible value for state $u$. This value depends on parameters $i,s,t$. It is specified by the corresponding stored path of the previous branch processing. Parameter $u$ of this branch is the path prefix that was stored in the previous branch processing after the comparison of the foursome metrics of paths ending in $'...is/t'$. Indeed, a path processed in this branch ending in $'...ist/...'$ is the extension of paths processed in the previous branch that end in $'...is/t'$. We recall from the previous branch processing that for each foursome paths ending in $'...is/t'$ (i.e. for each foursome paths of the previous branch's $t$ - th category and of a fixed pair $\{i,s\}$), only one path has survived. The prefix of this surviving path constitutes the prefix of all paths in this branch processing that end in $'...ist/...'$. For example, the first four metrics of this branch processing are:

$$\{M_{1000/0}, M_{2100/0}, M_{2200/0}, M_{1300/0}\} = \{-189.29, -49.02, -322.98, -160.68\}$$

with $M_{2100/0}$ being the survivor. Parameter $u$ in the above metrics was chosen from the corresponding stored paths of the previous branch. For instance, for $\{s,t\} = \{0,0\}$ and $i = 3$ the prefix state $u$ is chosen equal to 1 $(M_{1300/0})$. This is because $'130/0'$ was the surviving path of the fourth comparison in the previous branch processing:

$$\{M_{030/0}, M_{130/0}, M_{230/0}, M_{330/0}\} = \{-471.62, -101.53, -139.31, -183.23\}$$

The metrics of (3.25) are calculated for all paths of the first category. For each of the 16 pairs of $\{s,t\}$, four metrics are compared and only one along with its path survives. That concludes comparisons and storing of the first category of paths. The same procedure is executed for the rest three categories, i.e. for paths $uist/1$, $uist/2$ and $uist/3$. All metrics are computed recursively with the use of metrics $M_{uis/t}$ stored in the previous branch processing. For example, a path of the fourth category is computed as

$$M_{uist/3} = M_{uis/t} - \left(y_3 - \left(f_{-1}s_4 + f_0 s_3 + f_1 s_2\right)\right)^2 \qquad (3.26)$$

Branches 4, 5, 6

In processing the next branches up to branch 6, the same procedure of branch 3 is followed. Differentiations exist only in the last $(C-1)$ branches, i.e. the branches that contain the resetting zero input bits. In our case, differentiations exist only in the last branch, branch 7. In processing branch $b,\ 4 \le b \le 6,$ decoder firstly computes the following four path metrics

$$M_{\underbrace{....ui00/0}_{b+2}} = -\sum_{k=0}^{b}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 \quad i = 0,1,2,3 \tag{3.27}$$

It likewise chooses the one with the largest value. This is naturally fulfilled for every possible value of the last three zero path suffixes $('...00/0')$. Hence, we come up again with a total of 64 surviving paths and metrics that are stored at the end of each branch. Prefix $u$ and all other prefixes before it, are similarly indicated from the relevant stored path of the previous branch. For instance, for the first four path metrics of (3.27), parameter $u$ and all other prefixes before it, are exactly the prefixes of the only one stored path of the previous branch processing that ends in $'\underbrace{...i0/0}_{b+1}'$.

Branch 7

Branch 7 is the last branch of this example and there are no next $B_F$ branches to be considered. The number of computations and stored path metrics is decreased by a factor of $2^k = 4.$ 64 paths and their metrics from the processing of the previous branch are received. Since no next branch exists, no path extension is performed. One more Euclidean distance is added to each one of the 64 metrics and 64 new metrics are produced. This Euclidean distance corresponds to the likelihood of the last received symbol $(y_7)$ to be interpreted correctly. The 64 new calculated path metrics are sorted to groups of four, depending on the last two branch transitions $\{s,t\}.$ In contrast to the processing of previous branches, there are no categories of transitions of next "interfering" branches. For every pair of ending states $\{s,t\}$ the group's metrics are calculated as

$$M_{\underbrace{....uist}_{8}} = -\sum_{k=0}^{7}\left( y_k - \sum_{j=-1}^{1} f_j s_{k-j} \right)^2 \quad i = 0,1,2,3 \tag{3.28}$$

Prefixes before $i$ are similarly taken from the corresponding survived path of branch 6 processing. For each group, only one path survives, so that 16 surviving paths and metrics are extracted. For example, the metrics of the first four groups $(s = 0,1,2,3$ and $t = 0)$ are:

$$\{M_{22103000}, M_{22211100}, M_{22211200}, M_{22130300}\} = \{-68.70, -32.82, -115.79, -36.42\}$$

$$\{M_{22212010}, M_{22131110}, M_{10223210}, M_{22130310}\} = \{-23.27, -10.65, -87.84, -0.21\}$$

$$\{M_{22212020}, M_{22103120}, M_{10223220}, M_{22211320}\} = \{-61.34, -113.558, -81.10, -164.85\}$$

$$\{M_{22103030}, M_{22131130}, M_{22211230}, M_{22130330}\} = \{-49.04, -12.28, -24.14, -19.10\}$$

Consequently, 22211100, 22130310, 22212020, 22131130 are the survivors for the first four groups. 16 more metric computations take place for each of the other three path groups defined for $t = 1,2,3.$

From the 16 resulting survivors, decoder focuses only on the first four, mentioned above. The reason is that only survivors from the first four groups end in state 0. Decoder "is aware" that the input bits of the last branch were zero and takes advantage of this knowledge in order to choose the final optimum path. In other words, the input zero bits of the last branch are predefined and known to the receiver. Nevertheless, decoder still processes branch 7 in order to exploit the information contained in the value of received symbol $y_7$.

The final decision on the transmitted sequence is made between the first four survivors. Path $22130310$ exhibits the largest metric. It is the final chosen path. Transformation into input bits is conducted easily with the help of the Trellis:

$$\{1,0,1,0,0,1,1,1,0,0,1,1,0,1,0,0\}$$

This sequence is the same with the originally transmitted bit sequence. Decoder has therefore successfully interpreted received corrupted information $y_k$ of (3.20) into the transmitted digital information, with no errors. In this example, the channel was assumed to be precisely known beforehand. As said, in reality channel's response coefficients change over time and are estimated with adaptive techniques. That makes interpretation for all decoders more difficult. However, if we can prove that for exact channel knowledge, MS decoder produces fewer errors than other optimum standard equalising and decoding schemes (working also with exact channel knowledge), then it is understandable that MS decoder will provide better results when adaptive methods are used in both cases too.

In practice, the length of the transmitted bursts is quite longer. It usually fluctuates around a few hundreds or even a few thousands of bits, depending on the application. The example assumed very short segments so that path notation would be feasible. Whatever the length of the burst, operation principles described above remain the same. Another observation to be made is the likeness of all paths the more branches we go back in time. As can be seen from the paths of the first four groups above, many paths "tend" to be similar in their beginning part. The first three branch transitions of many paths are the same. In fact, there exist only three different path starts concerning the first three branches. If we studied much longer bursts we would notice that the start of all surviving paths is the same, many branches back in time. Path similarity observed in this short example, is a good indication of the relevant principle. As mentioned in section 3.1, a length of $5C$ or $5L$ branches back in time would enable us to make correct path choices, with the insertion of a decoding delay.

## 3.4 Practical consideration

### 3.4.1 Basic use

We focus on ways and practical schemes with which MS decoding can be exploited. First of all, we consider the basic employment scenario. MS decoder is a device that substitutes every set comprised of an equaliser and a following convolutional decoder. The inexistence of interleaving between these two devices, presupposes communication structures where a convolutional encoder is situated exactly before modulation inside the transmitter. Such a scheme should employ devices shown in Fig. 3.4-1.

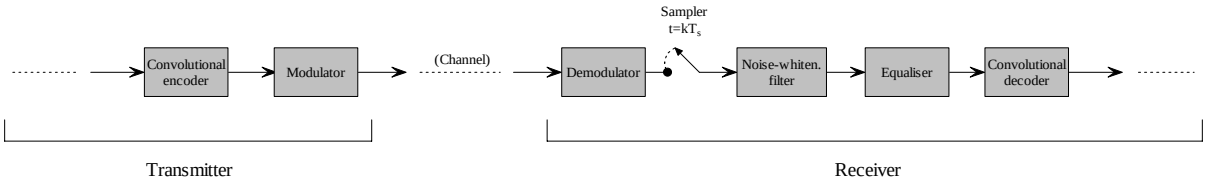


**FIGURE 3.4-1**
Standard technique (scheme 1).

The prerequisite is that there exists a serial concatenation of a convolutional encoder and a modulator at the end of the transmitter. The modulated signal is transmitted through the channel and demodulated, potentially with the help of matched filter. It is afterwards digitised, and passed through the noise whitening filter, an equaliser and a convolutional decoder. Other kinds of devices may follow convolutional decoder. Correspondingly, other kind of operations can be conducted before convolutional encoding in the transmitter. All of these pose no disturbance as far as usage of MS decoder is concerned. Equaliser and convolutional decoder are simply substituted by MS decoder as shown in Fig. 3.4-2.

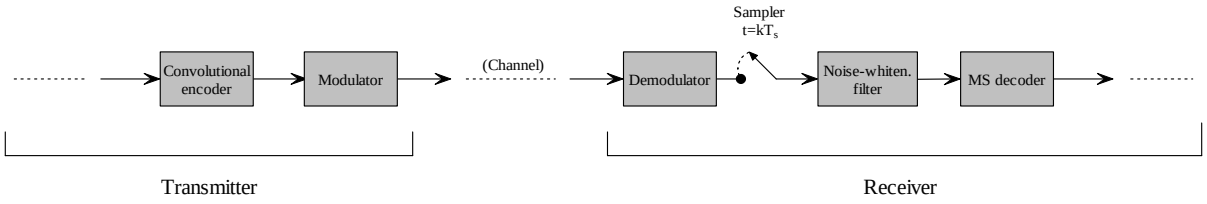


**FIGURE 3.4-2**
Employing MS decoder (scheme 2).

Let us see some clues concerning the error performance of the afore mentioned schemes. For a given signal to noise ratio per bit $E_b / N_0$, the bit error percentage is estimated. Ratio $E_b / N_0$ represents the analogy between signal power and noise power. Noise assumptions are therefore included in this ratio. Interference assumptions are specified by the equivalent discrete model. In order to cover a variety of possible scenarios, we will examine the bit error rates $(BER)$ for four different interference models. Every model corresponds to a different ISI category, in ascending scale. The first model represents a good quality telephone channel, the second introduces medium to heavy amount of interference and the third severe. The fourth model tallies as one of the worst case scenarios; it is usually met in corrupted wireless or underwater acoustic channels. The impulse responses are shown in Fig. 3.4-3 to 3.4-6.

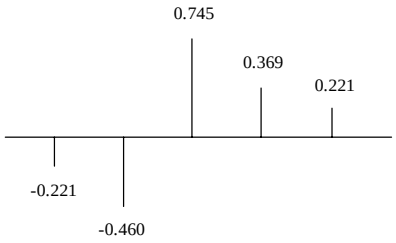


**FIGURE 3.4-3**
Equivalent discrete model 1.

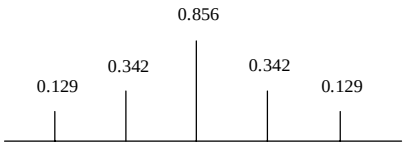


**FIGURE 3.4-4**
Equivalent discrete model 2.

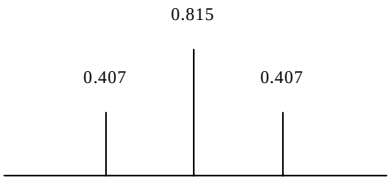


**FIGURE 3.4-5**
Equivalent discrete model 3.

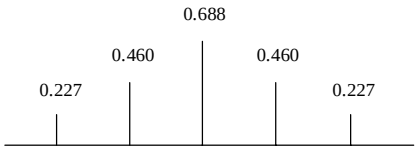


**FIGURE 3.4-6**
Equivalent discrete model 4.

Figures 3.4-7 to 3.4-10 display the performance of schemes 1 and 2 for each of the four equivalent discrete models. Selectivity on $E_b / N_0$ causes differentiations in $BER$. Bit error rates are calculated based on the number of different bits between the input of the convolutional encoder and the output of the optimum Viterbi decoder for scheme 1 or MS decoder for scheme 2. $64$ - QAM modulation, MLSE for separate equalisation and the convolutional encoder of Fig. 2.3-1 are used. $64$ - QAM modulation implies $8$ - PAM baseband modulation. The channel is assumed to be known to the receiver and hence no adaptive techniques were used to estimate the coefficients of any of the four equivalent discrete models.

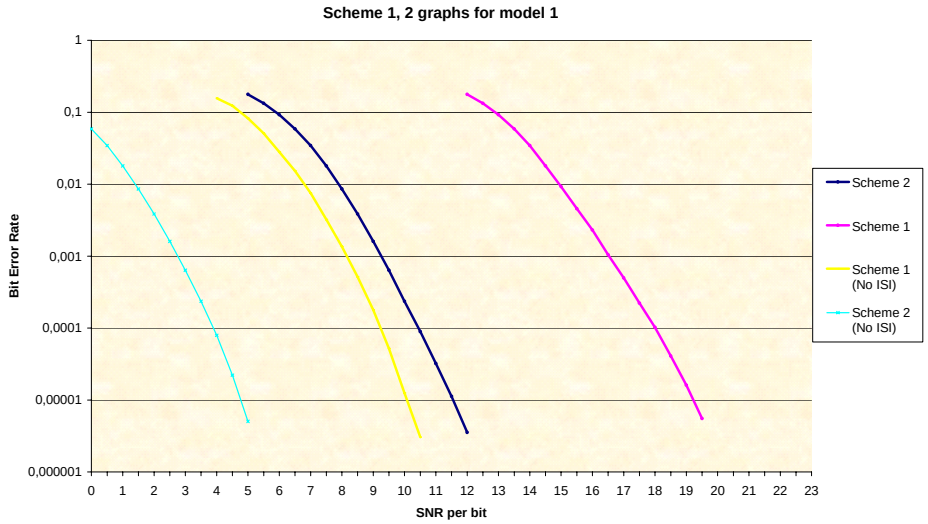


**FIGURE 3.4-7**
Scheme 1 and 2 performance for eq. discrete model 1, $8$ - PAM and $(2,2,3)$ conv. coding.

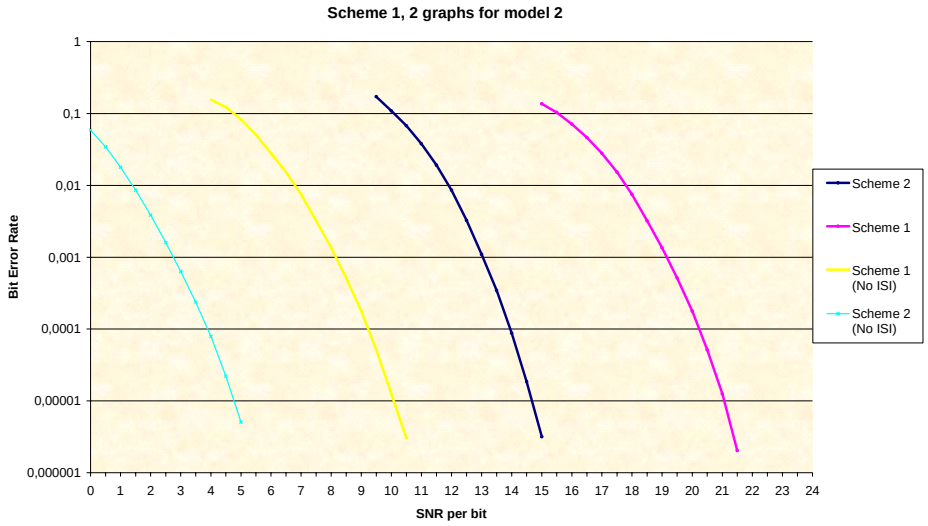


**FIGURE 3.4-8**
Scheme 1 and 2 performance for eq. discrete model 2, $8$ - PAM and $(2,2,3)$ conv. coding.

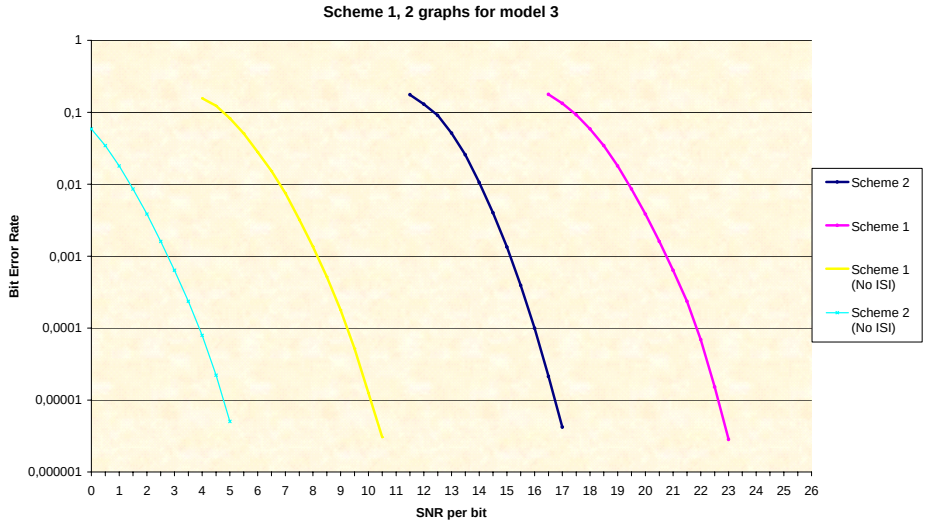


**FIGURE 3.4-9**
Scheme 1 and 2 performance for eq. discrete model 3, $8$ - PAM and $(2,2,3)$ conv. coding.

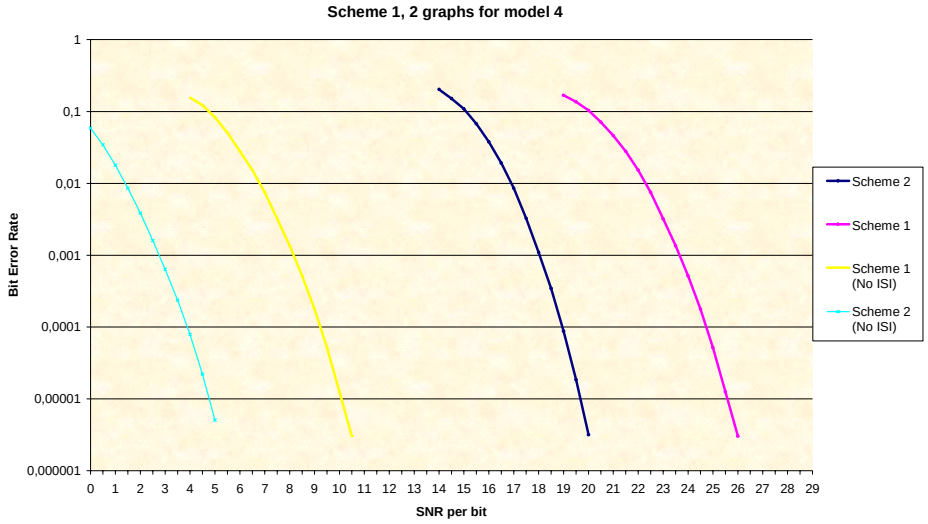


**FIGURE 3.4-10**
Scheme 1 and 2 performance for eq. discrete model 4, $8$ - PAM and $(2,2,3)$ conv. coding.

For comparison reasons, the ideal case of no intersymbol interference is also pictured on the graphs, for each of the tested schemes. As can be seen, there is a significant improvement in all cases. The ordinary concern is to achieve a $BER$ less than $10^{-4}$, depending of course on the application. If we look at the area where $BER \approx 10^{-5}$ we see that a minimum of approximately $6$ dB is saved on power for equivalent discrete model 4. For the same error rates, reduction reaches up to $7.5$ dB in the case of model 1 and in between for the rest two channel models. For model 1, we see that the obtained results are $1.5$ dB away from the scheme 1 simulation results with no ISI. General improvement is evident when comparing the 'No ISI' cases of each scheme as well. As mentioned, in practice channel and model coefficients vary over time. Consequently, comparison of the 'No ISI' cases is useful from a practical point of view.

### 3.4.2 Universal use without bandwidth expansion

Schemes 1 and 2 shown in Fig. 3.4-1 and 3.4-2 require that no interleaving is used between convolutional encoder and modulator. In many telecommunication standards nowadays, this is not satisfied. Interleaving is indispensable when dealing with occurrences such as multipath fading and signal loss [22], [23]. Typically, an inteleaver is employed at the output of a convolutional encoder before modulation. Interleavers are much more effective when fed with bit or symbol sequences that contain redundancy.

In order to make MS decoding possible for all cases without increasing the channel's required bandwidth, we substitute devices in Fig. 3.4-11 by devices of Fig. 3.4-12. We name the two block diagrams "Scheme 3" and "Scheme 4" respectively. The $4$- th order modulator of scheme 3 is replaced by a $1/2$ rate convolutional encoder and a $16$- th order modulator in scheme 4. Higher modulation is used. MS decoder, instead of MLSE, processes the sampled output of the demodulator. With the use of higher modulation, we make sure that for a certain input bit rate the same symbol rate is required in both cases. Maintaining the same symbol rate ensures that no additional bandwidth is needed, since these two quantities are connected. Naturally, scheme 4 employs higher baseband modulation too. $4$- level PAM instead of binary PAM is used.

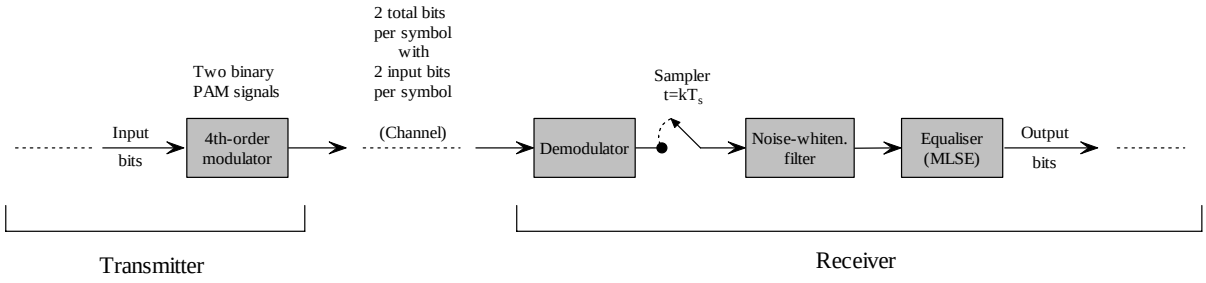


**FIGURE 3.4-11**
Standard technique (scheme 3).

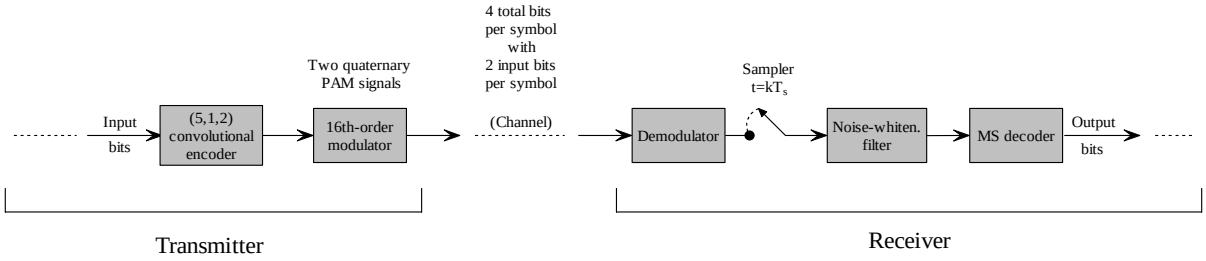


**FIGURE 3.4-12**
Employing MS decoder without increasing the bandwidth (scheme 4).

The $4$-th order modulator can be of any type of modulation technique, such as DQPSK, GMSK etc. $16$-th order modulator is $16$-PSK. $16$-QAM can also be considered under special conditions [25], [26]. Fig. 3.4-13 to Fig. 3.4-16 display the performance of schemes 3 and 4 for the four equivalent discrete models. The $(5,1,2)$ convolutional encoder of Fig. 3.4-17 is used in the transmitter of scheme 4 and an ML detector performs equalising in the receiver of scheme 3. Encoder of Fig. 3.4-17 has good error correction properties $(d_{FD} = 7)$ and it is also employed within channel coding of GSM standard, as will be seen later.

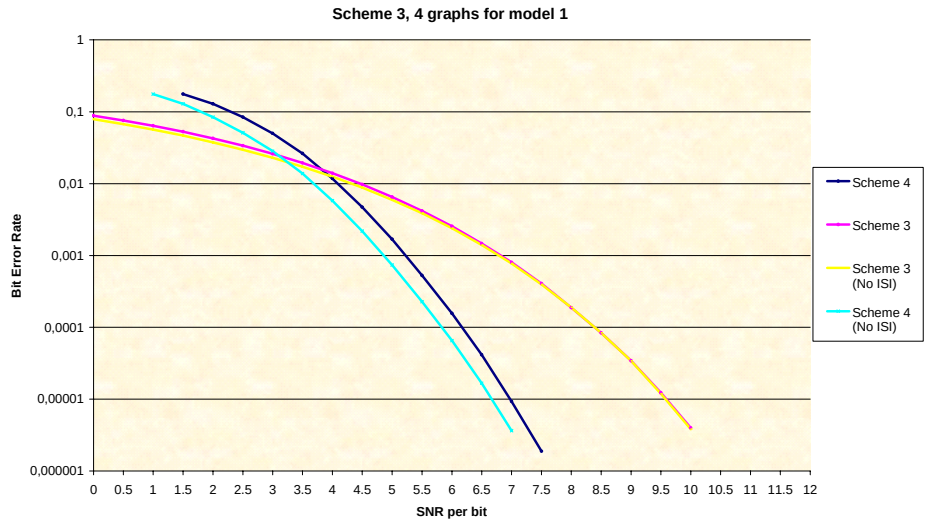


**FIGURE 3.4-13**
Scheme 3 and 4 performance for eq. discrete model 1.

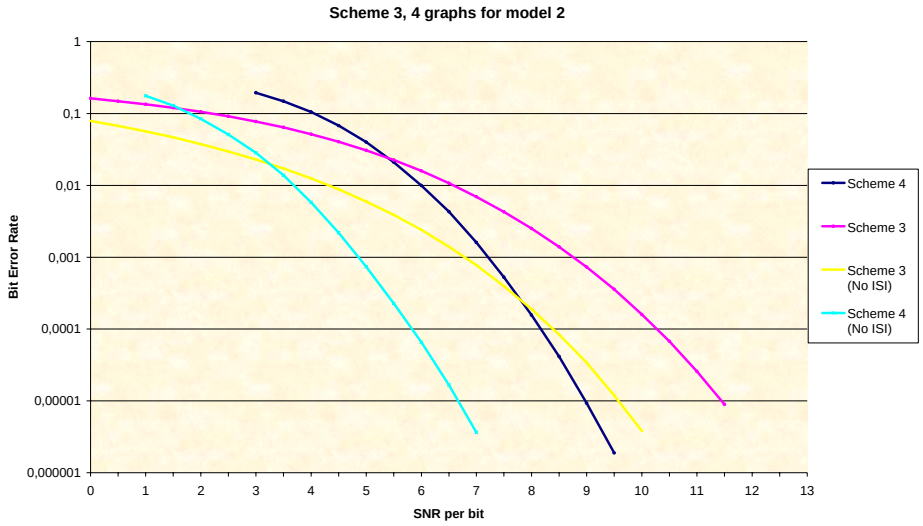


**FIGURE 3.4-14**
Scheme 3 and 4 performance for eq. discrete model 2.

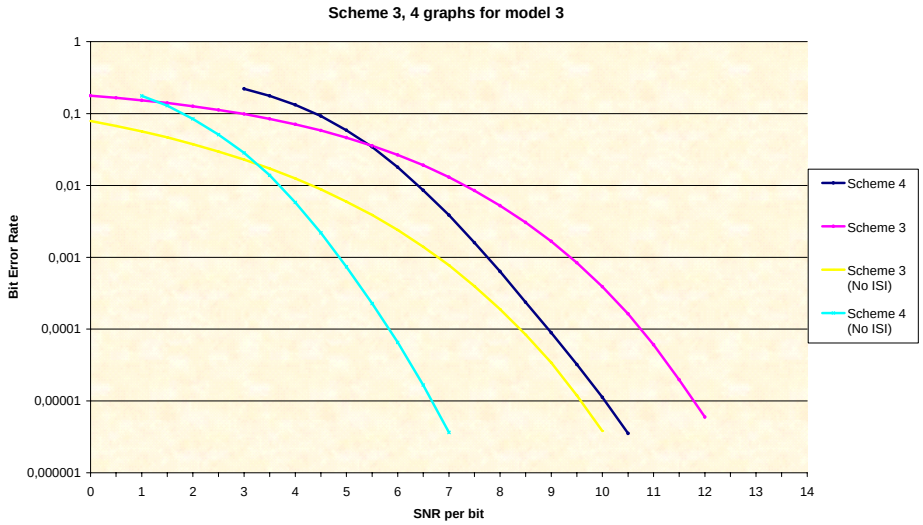


**FIGURE 3.4-15**
Scheme 3 and 4 performance for eq. discrete model 3.

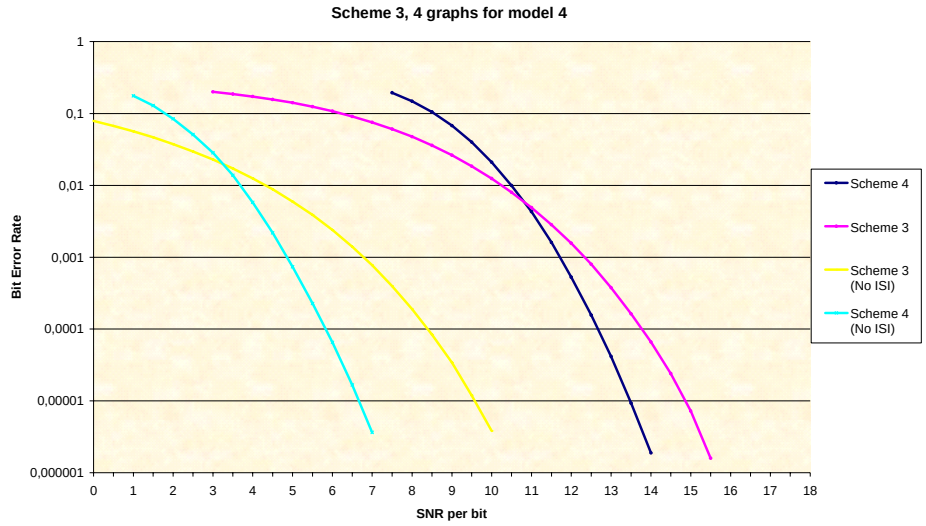


**FIGURE 3.4-16**
Scheme 3 and 4 performance for eq. discrete model 4.

Generally, we note that in spite of using increased modulation order, scheme 4 yields better results than scheme 3. The more interference exists, the less power reduction is achieved. However, even in the case of eq. discrete model 4 (where severe ISI is involved), a considerable reduction of $1.4\,\text{dB}$ is accomplished concerning a $10^{-5}$ bit error rate. In the best cases, improvement approaches $2.6\,\text{dB}$ as shown in Fig. 3.4-13, always referring to an error level of $10^{-5}$. A noteworthy remark is the loss of performance caused by serious ISI. In Fig. 3.4-13, the channel is relatively good and ISI poses no problem, although it is created by four interfering coeffiecients $(L+F=2)$. Scheme 3 performance is approximately the same with the performance of the 'No ISI' scenario for model 1. Picture changes dramatically for extremely bad-quality channels such that of model 4. As seen in Fig. 3.4-16, ISI presence results in a $5\,\text{dB}$ loss. A part of it $(1.5\,\text{dB})$ can be compensated through employment of scheme 4.

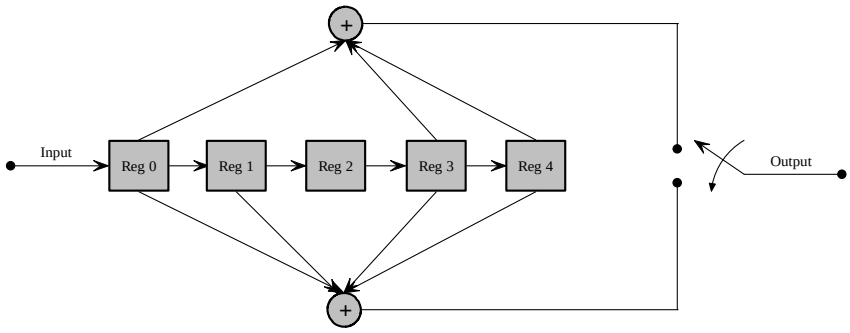


**FIGURE 3.4-17**
$C=5,\ k=1,\ n=2$ convolutional encoder.

It is important to note that it is not mandatory to employ particular convolutional coding and modulation in order to use MS decoder. Various kinds of encoders and modulators can be used. The case of schemes 3 and 4 was analysed because it is one of the most typical cases of employing MS decoding with a benefit. If we want to insert MS decoder in a standard, then a modification such that of scheme 4 could be made. Assuming that a standard normally utilises $M$ - level baseband signalling, the signalling level used after the modification would be equal to or greater than $M^{n/k}$ where $n/k$ is the reverse of the code rate of the convolutional encoder used within the modified standard. For example, a modulator using $4$ - level PAM signalling can be replaced by a $2/3$ rate convolutional encoder and a following modulator using $4^{3/2}=8$ - level PAM signalling. In general, employment of MS decoder without a bandwidth increase is not recommended for particularly high levels of signalling. The reason is that power requirements of very high modulation schemes increase significantly for every $M$ increment and a dB improvement similar to that of Fig. 3.4-13 to 3.4-16 is not feasible.

### 3.4.3 Trellis-coded modulation

As mentioned, employment of MS decoder necessitates the existence of a convolutional encoder before modulator in the transmitter. Because of this existence, dependencies are introduced between the modulated symbols. MS decoder takes advantage of the dependencies in order to correct errors. A question arises at this point. Why don't we use these dependencies in order to construct proper modulation that would resist to errors more tenaciously? In other words, why don't we exploit Trellis knowledge so as to build a mapping which would help MS decoding? We need first make sure that the symbol values corresponding to Trellis transitions emanating from the same state or leading to the same state would distance themselves as much as possible. Then we would try to maximise the difference between the Euclidean distance properties of the shortest Trellis paths emanating from and merging to the same state.

In this case, symbol mapping would be designed to fit the needs of the preceding convolutional encoder. This is the key feature of a modulation technique invented by Ungerboeck in 1982 [27]. It is called *Trellis-coded modulation* (TCM). It can be used within a communication structure in combination with MS decoder. The distinctive mapping used, is called *set partitioning*. With the use of set partitioning, MS decoding results are significantly improved.

For the needs of this thesis, we will focus on set partitioning of PAM symbols. Each of the $M$ - level PAM symbols corresponds to $\log_2 M$ output bits of a Trellis transition. Our aim is to map the symbols in such a way that $d_{FED}$ is maximised. In order to achieve that, we take special care that symbols corresponding to transitions that converge to the same state are sorted in a most distant way. The same applies for symbols corresponding to transitions that originate from the same state. Let us study the example of Trellis in Fig. 2.3-2. We assume that $M=8$ and thus each branch transition corresponds to one baseband symbol. We will refer to the 3-bit output of each transition as "output symbol". Each group of three different bits has to be assigned a symbol value from the set {-7, -5, -3, -1, +1, +3, +5, +7}. We note that four transitions emerge from any state and naturally, four transitions converge to any state. Therefore, we begin by selecting the values for the four output symbols of transitions leading to state 0. The values are spaced as much as possible and equally likely, keeping a distance of "4" between them. We select values -7, -3, +1, +5 for the output symbols 101, 000, 011, 110 respectively. We notice that the four remaining values correspond to the output symbols of transitions leading to state 1. Hence, values -5, -1, +3, +7 are distributed to the output symbols 100, 001, 010, 111. We observe that the four output symbols of transitions leading to state 2 are the same with the four output symbols of transitions leading to state 1. They are assigned the same values. Equally, the four output symbols of transitions leading to state 3 are the same with the four output symbols of transitions leading to state 0. We succeeded in keeping a minimum distance of "4" among output symbols of converging transitions. That is the maximum we can do. Next, based on the already accomplished distribution, we check the distances of output symbols of transitions emanating from the same state. We see that for states 0 and 2 a minimum distance of "4" is also maintained.

Unfortunately, this is not satisfied for states 1 and 3. Output symbols 001 and 011 exhibit a distance of "2" (-1, +1). Consequently, the minimum spacing of output symbols of all transitions merging to the same node or emanating from the same node is two and $d_{FED} = 20.$ Whatever different mapping we attempt, $d_{FED}$ cannot be set bigger than $20.$ We choose this mapping as the final one. It is displayed in Fig. 1.3-2.

By coincidence, this is also a Gray-coded mapping. Generally, when it comes to MS decoding, mapping with the method of set partitioning is strongly preferred. Ungerboeck devised three heuristic rules so that the best TCM mapping can be achieved in any case:

1. Use all subsets of symbol values with equal frequency in the Trellis.
2. Output symbols of transitions emanating from the same state or merging into the same state in the Trellis are assigned subsets of symbol values that are separated by the largest Euclidean distance.
3. If parallel transitions exist, they are assigned subsets of symbol values that are separated by the largest Euclidean distance.

With the term "parallel" he described transitions caused by uncoded bits. A typical example is shown in Fig. 3.4-18. This is one of the most known codes from Ungerboeck [28]. Its rate is $k/n = 2/3$ and consequently its output consists of three bits. The first output bit is simply the first uncoded input bit. The second and third output bits are the output of an inner $(3,1,2)$ convolutional encoder whose input is the second uncoded bit. The corresponding Trellis is displayed in Fig. 3.4-19.

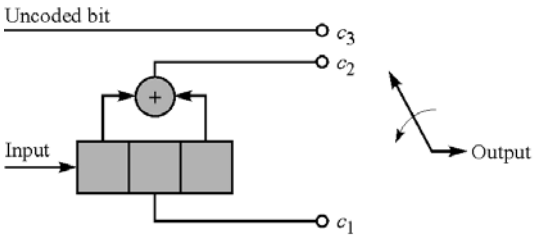


**FIGURE 3.4-18**
An encoder used with TCM.

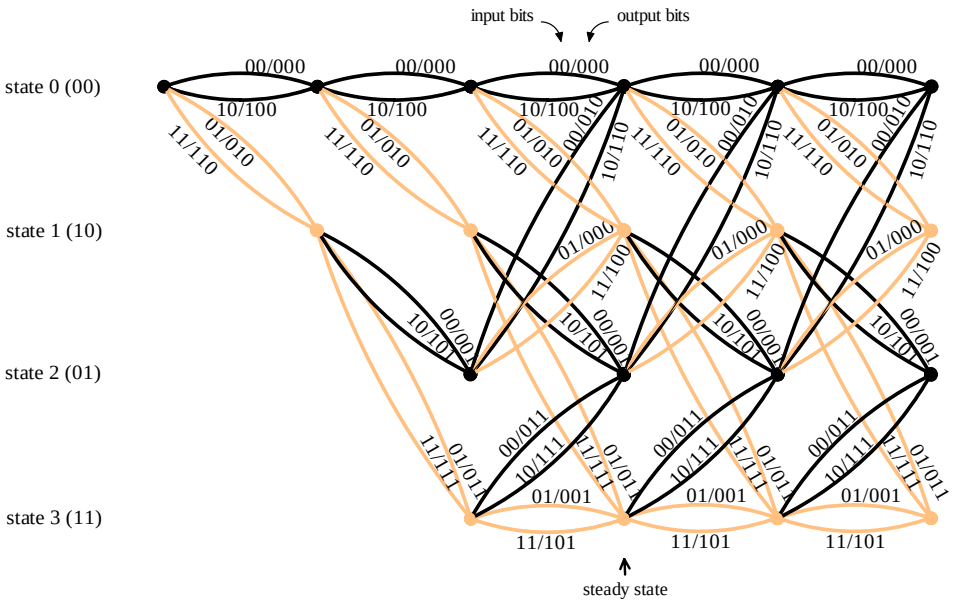


**FIGURE 3.4-19**
Trellis diagram for encoder of Fig. 3.4-18.

The above encoder is used to obtain simulation results in the next chapter. Let us study the PAM mapping that would enable us to make use MS decoder in reception more effectively. Looking at the Trellis in Fig. 3.4-19, we notice a difference in comparison with the previous Trellis diagrams. There exist different transitions originating and ending to same states. These are the parallel transitions. Their output bits differ only with respect to the uncoded bit or bits. In our case we have to do with one uncoded bit. Hence, there is a maximum of two parallel transitions for every pair of a starting and an ending state.

We begin with Ungerboeck's first rule. Indeed, output symbols of all transitions are used with the same frequency. Each one of the 3-bit output symbols is used twice in a Trellis branch. In order to comply with the second rule, we act similar with the previous example. For the group of four output symbols 000, 110, 100, 010 of transitions ending either in state 0 or in state 1, we assign the values -7, -3, +1, +5 respectively. Each pair of these output symbols is spaced by a minimum Euclidean distance of 16. The four values -5, -1, +3, +7 are given likewise to the four left output symbols 001, 011, 101, 111 of transitions ending either in state 2 or in state 3. They too are separated by a minimum square distance of 16. What remains to be satisfied is now the third rule. Each pair of output symbols differing only in the first output bit have to have a minimum distance of eight, i.e. a minimum Euclidean distance of 64. A greater minimum Euclidean distance for all pairs of symbols is not possible. Fortunately, we observe that the distribution we made so far, satisfies this criterion. If it did not, we should go back and redistribute symbol values. In that case, the redistribution would have to ensure that both the second and the third rule are satisfied. The mapping procedure has been completed. It is shown in Fig. 3.4-20. If we look at the Trellis we will see that a free Euclidean distance $d_{FED} = 36$ has been achieved. On the other hand, parallel transitions cause hardly any problem. They are separated by a minimum Euclidean distance of 64. MS decoder will have an easier task in order to decide which one of the parallel transitions is to be rejected.

| | |
|---|---|
| $"111" \rightarrow 7A$ | $"011" \rightarrow -A$ |
| $"010" \rightarrow 5A$ | $"110" \rightarrow -3A$ |
| $"101" \rightarrow 3A$ | $"001" \rightarrow -5A$ |
| $"100" \rightarrow A$ | $"000" \rightarrow -7A$ |

**FIGURE 3.4-20**
8-PAM mapping to be used for MS decoding of the encoder of Fig. 3.4-18.

# 4. MS DECODING PERFORMANCE

## 4.1 Probability of error

In this section we shall calculate the error probability of MS decoder when the additive noise is Gaussian. As usual, we assume a $(C,k,n)$ convolutional encoder and $M$ - level PAM modulation in the transmitter, with $n$ equal to or a multiple of $\log_2 M$. The channel introduces ISI from $F$ following and $L$ preceding symbols. The calculated bit error probability $P_e$ has to be a function of the signal's power. The more power we use in a transmission, the fewer errors we have. These two factors are inversely proportional. The main parameter used to represent power in telecommunications is *average signal to noise ratio per bit*, $E_b/N_0$. $E_b$ is the *average signal energy per bit*. $N_o$ is called *one-sided noise power spectral density (psd)* and is the derivative of average noise power towards bandwidth.

Knowing that $M$ - level PAM is employed, we understand that $T_s = T_b \log_2 M$ where $T_s$, $T_b$ are the symbol and bit durations respectively. Therefore, we have

$$f_s = \frac{f_b}{\log_2 M} \tag{4.1}$$

with $f_s$ denoting the symbol rate and $f_b$ the bit rate. According to Nyquist [29], the required bandwidth $B_n$ in order to receive the entire information from a source emitting $f_s$ symbols per second, is

$$B_n = \frac{f_s}{2} \tag{4.2}$$

Moreover, with the help of basic physics we estimate

$$\frac{E_b}{N_0} = \frac{ST_b}{N_0} = \frac{ST_bB_n}{N} \tag{4.3}$$

where $S$, $N$ are the average signal and noise power respectively. Equation (4.3) with the help of (4.1) and (4.2) yields

$$\frac{E_b}{N_0} = \frac{1}{2\log_2 M}\frac{S}{N} \tag{4.4}$$

where $S/N$ is the *average signal to noise ratio*. In practice, the actual bandwidth is a little bigger than $f_s/2$ for baseband signalling. It is $f_s(1+\beta)/2$ where $0.19 \le \beta \in [0,1]$ is the matched filter's roll-off factor (section 1.2). This however, does not make any difference as far as theoretical derivation of (4.4) is concerned.

For an $M$ - level PAM signal, the average signal power is

$$S = \frac{1}{M}\sum_{m=-M/2}^{M/2}\left(2|m|-1\right)^2 A^2 = \frac{2}{M}\sum_{m=1}^{M/2}\left(2m-1\right)^2 A^2 \Rightarrow S = \frac{M^2-1}{3}A^2 \tag{4.5}$$

where $m \in Z^*$ and $2A$ is the voltage amplitude distance between two successive PAM signalling levels.

We will denote with $s_k \quad k = 0,1,2,...$ the sequence of the transmitted PAM symbols. We will use notation $\hat{s}_k \quad k = 0,1,2,...$ for the transmitted PAM symbol sequence estimated by MS decoder. These sequences correspond to the output bits showed in every branch of the Trellis diagram. In fact, if we set

$$p = \frac{n}{\log_2 M}, \quad p \in \aleph^* \tag{4.6}$$

then the output of each branch transition contains $p$ PAM symbols. We begin the analysis by assuming that an error takes place. The estimated path through the Trellis diverges from the correct path in branch $b$ and remerges with the correct path in branch $b+l, \quad b \in \aleph$ and $C-1 \le l \in \aleph.$ Hence, both paths pass through the same start-state of branch $b$ and through the same end-state of branch $b+l,$ but through different states in-between. We call this diversion an *error event*.

For such an error event, we understand that $\hat{s}_k$ is not equal to $s_k$ for all $k$ within the space $pb \le k \le p(b+l+1)-1.$ We define a corresponding error vector $e$ as

$$e = [e_{pb} \quad e_{pb+1} \quad \dots \quad e_{p(b+l+1)-1}] \tag{4.7}$$

with $e_k = 0, \ \forall k < pb$ and $\forall k > p(b+l+1)-1.$ The elements of $e$ are defined as

$$e_k = \frac{1}{2A}(s_k - \hat{s}_k), \qquad k = pb, pb+1, ..., p(b+l+1)-1 \tag{4.8}$$

Obviously, each element of the error vector takes on the values $0, \pm 1, \pm 2, \pm 3, ..., \pm(M-1).$ Furthermore, at least one element is not zero.

We are interested in determining the probability of error of the error event $E$ specified in equation (4.7). For $E$ to occur, the following two events $E_1$ and $E_2$ must occur:

$E_1$: The part of the total metric of the estimated path corresponding to branches $b, b+1, ..., b+l,$ must be larger than the part of the total metric of the correct path corresponding to branches $b, b+1, ..., b+l.$ The output bits of the transitions of these branches correspond to the transmitted symbols $s_{pb}, s_{pb+1}, ..., s_{p(b+l+1)-1}.$

$E_2$: The error sequence $2A(e_{pb}, e_{pb+1}, ..., e_{p(b+l+1)-1})$ when added to the correct symbol sequence $s_{pb}, s_{pb+1}, ..., s_{p(b+l+1)-1}$ must result in an allowable transmitted symbol sequence.

The probability of $E_1$ is

$$P(E_1) = P\left[ \sum_{k=pb}^{p(b+l+1)-1} \left( y_k - \sum_{j=-F}^{L} f_j \hat{s}_{k-j} \right)^2 < \sum_{k=pb}^{p(b+l+1)-1} \left( y_k - \sum_{j=-F}^{L} f_j s_{k-j} \right)^2 \right] \tag{4.9}$$

where $f_k$ are the coefficients of the equivalent discrete model and $y_k$ the received symbol sequence at the output of the noise-whitening filter. From (1.13) we have

$$y_k = \sum_{j=-F}^{L} f_j s_{k-j} + n_k \tag{4.10}$$

where $n_k$ is a Gaussian distribution of zero mean value and of variance $\sigma^2$. Substitution of (4.10) in (4.9) yields

$$P(E_1) = P\left[ \sum_{k=pb}^{p(b+l+1)-1} \left( n_k + 2A \sum_{j=-F}^{L} f_j e_{k-j} \right)^2 < \sum_{k=pb}^{p(b+l+1)-1} n_k^2 \right]$$

$$P(E_1) = P\left[ 4A \sum_{k=pb}^{p(b+l+1)-1} \left[ n_k \left( \sum_{j=-F}^{L} f_j e_{k-j} \right) \right] < -4A^2 \sum_{k=pb}^{p(b+l+1)-1} \left( \sum_{j=-F}^{L} f_j e_{k-j} \right)^2 \right] \tag{4.11}$$

If we define

$$d_k = \sum_{j=-F}^{L} f_j e_{k-j} \tag{4.12}$$

then (4.11) becomes

$$P(E_1) = P\left( \sum_{k=pb}^{p(b+l+1)-1} d_k n_k < -A \sum_{k=pb}^{p(b+l+1)-1} d_k^2 \right) \tag{4.13}$$

The first term in (4.13) is a linear combination of normally distributed values with weights $d_k$. These values represent white Gaussian noise since they are statistically independent according to the definition of the equivalent discrete model. The second term is a real negative number. Consequently, (4.13) is equivalent to:

$$P(E_1) = Q\left( \sqrt{\frac{A^2}{\sigma^2} \sum_{k=pb}^{p(b+l+1)-1} d_k^2} \right) \tag{4.14}$$

where $Q$ denotes the *complementary cumulative distribution function*:

$$Q(x) = \frac{1}{2} erfc\left( \frac{x}{\sqrt{2}} \right), \quad x \in \Re \tag{4.15}$$

and $erfc$ is the *complementary error function*:

$$erfc(x) = \frac{2}{\sqrt{\pi}} \int_{s=x}^{+\infty} e^{-s^2} ds, \quad x \in \Re \tag{4.16}$$

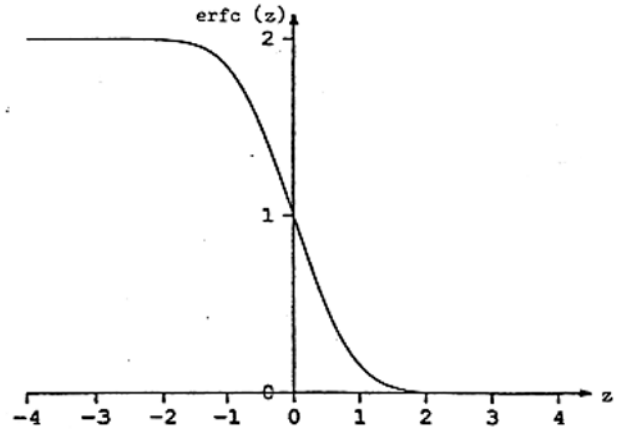


**FIGURE 4.1-1**
Complementary error function.

If we define the below positive real number for each error vector

$$\varepsilon(e) = \sum_{k=pb}^{p(b+l+1)-1} d_k^2 = \sum_{k=pb}^{p(b+l+1)-1} \left( \sum_{j=-F}^{L} f_j e_{k-j} \right)^2 \tag{4.17}$$

then with the help of (4.4) and (4.5), equation (4.14) yields

$$P(E_1) = Q\left( \sqrt{\frac{6\varepsilon(e)\log_2 M}{M^2 - 1} \frac{E_b}{N_0}} \right) \tag{4.18}$$

We have used equality $N = \sigma^2$, i.e. the average noise power is equal to the variance of the normally distributed noise sequence $n_k$.

Equation (4.18) is very useful in order to make some observations. First, we verify that the bigger the constraint length $C$ is, the bigger $l$ and $\varepsilon(e)$ are and consequently, the smaller the probability of error is. As expected, we notice that a code with larger free Euclidean distance $d_{FED}$ causes more non-zero error elements $e_k$ and that increases $\varepsilon(e)$ and decreases $P(E_1)$. Next, we observe that in the absence of intersymbol interference, $\varepsilon(e) \approx 1$, and $P(E_1)$ is proportional to the symbol error rate of $M$ - level PAM. Another thing we verify is that susceptibility to errors affects mostly neighbouring symbols. For symbols that are not mapped in neighbouring positions, $|e_k|$ is greater than 1 (+2, +3, …etc). $\varepsilon(e)$ is exponentially increased and probability of error decreases significantly, approaching to 0.

With our calculations so far, we computed the symbol error probability considering every possible case of $e_k$ taking on the values $0, \pm 1, \pm 2, \pm 3, ..., \pm(M-1)$. In some particular cases however, the value of $e_k$ generates illegal values for the pair $\{s_k, \hat{s}_k\}$. For example, for $e_k = M - 1$ and $s_k = 1$ an unallowable value is calculated for $\hat{s}_k$ through (4.8). Although mathematically these particular cases are taken into account within equation (4.18), decoder does not consider them in practice.

Furthermore, even if both values of $\{s_k, \hat{s}_k\}$ are legal, MS decoder still neglects some cases. As we remember from the previous chapter, not all combinations of transmitted symbols $s_k$ exist over the Trellis paths. This limitation is used to correct errors and distinguishes MS decoder results from the MLSE. As a result, event $E_2$ can be decomposed into the following two sub-events $E_2^a$ and $E_2^b$:

$E_2^a$: The values of sequences $\hat{s}_{pb}, \hat{s}_{pb+1}, ..., \hat{s}_{p(b+l+1)-1}$ and $s_{pb}, s_{pb+1}, ..., s_{p(b+l+1)-1}$ must belong to the set {±A, ±3A, ±5A, ..., ±(M-1)A}.

$E_2^b$: The values of the candidate sequence $\hat{s}_{pb}, \hat{s}_{pb+1}, ..., \hat{s}_{p(b+l+1)-1}$ must correspond to an existing Trellis path.

The probability of $E_2^a$ depends on the statistical properties of $s_k$. We assume that the elements of $s_k$ are equally probable. Then for an error of the form $|e_k| = k,\ k = 1,2,...,M-1,$ and for a given legal $\hat{s}_k$ value, there are $M-k$ possible values of $s_k$ such that

$$s_k = \hat{s}_k + 2Ae_k \tag{4.19}$$

Therefore we have

$$P(E_2^a) = \prod_{k=0}^{p(l+1)-1} \frac{M - |e_k|}{M} \tag{4.20}$$

with $|e_0| = 0.$ Note that with the assumption that $\hat{s}_k$ has a permitted value, we do not lose generality.

The probability of $E_2^b$ depends on the properties of the convolutional code. We recall from the error event definition that the incorrect path diverges from the correct path in branch $b$ and remerges with it in branch $b+l$ with $l \geq C-1.$ The transitions of the two paths in these $l+1$ branches are different. Starting from the start-state of branch $b,$ there are $2^k$ existing transition choices in every branch out of a total $2^n$ $\hat{s}_k$ values. Hence, a ratio of $2^{n-k}/2^n$ choices are not examined by MS decoder for each one of the $l+1$ branches. We therefore estimate

$$P(E_2^b) = \prod_{i=b}^{b+l} 2^{(k-n)} = 2^{(l+1)(k-n)} \tag{4.21}$$

(4.21) was estimated with the prerequisite that the transmitted symbols are equally probable. We note for one more time the positive influence that an increase of $C$ has over the probability of error, at the cost of greater decoding complexity. In addition, an increase in $n-k$ reduces $P(E_2^b)$ too, at the cost of more power or bandwidth depending on the modulation used.

Because of the above, we conclude that the probability of the error event $E$

$$P(E) = P(E_1 \cap E_2^a \cap E_2^b) = P(E_1)P(E_2^a)P(E_2^b) \tag{4.22}$$

is given by

$$P(E) = 2^{(l+1)(k-n)} Q\left(\sqrt{\frac{6\varepsilon(e)\log_2 M}{M^2-1}\frac{E_b}{N_0}}\right) \prod_{k=0}^{p(l+1)-1} \frac{M - |e_k|}{M} \tag{4.23}$$

Let $T$ be the set of all error events $E$ that start from branch $b$. For each error event $E \in T$ let $d(E)$ be the Hamming distance between the input bits of the correct path and the input bits of the selected path. The average error probability concerning the input bits of the convolutional encoder preceding modulator is

$$P_e = \sum_{E \in T} d(E) P(E) \tag{4.24}$$

With the help of (4.23) we notice that (4.24) is a multiplication of four factors. We focus on the third factor which contains the signal to noise ratio. Let $\Lambda$ be the set of all $\varepsilon(e)$. For every $\varepsilon \in \Lambda$, let $\Lambda_E \subseteq T$ be the subset of error events for which $\varepsilon(e) = \varepsilon$. We have:

$$P_e = \sum_{\varepsilon \in \Lambda} \left[ Q\left( \sqrt{\frac{6\varepsilon \log_2 M}{M^2 - 1} \frac{E_b}{N_0}} \right) \sum_{E \in \Lambda_E} \left( d(E) 2^{(l+1)(k-n)} \prod_{k=0}^{p(l+1)-1} \frac{M - |e_k|}{M} \right) \right] \tag{4.25}$$

Rewriting the above we get the final expression:

$$P_e = \sum_{\varepsilon \in \Lambda} C_\varepsilon Q\left( \sqrt{\frac{6\varepsilon \log_2 M}{M^2 - 1} \frac{E_b}{N_0}} \right) \tag{4.26}$$

where

$$C_\varepsilon = \sum_{E \in \Lambda_E} \left( d(E) 2^{(l+1)(k-n)} \prod_{k=0}^{p(l+1)-1} \frac{M - |e_k|}{M} \right) \tag{4.27}$$

As expected, if no convolutional encoder is used before modulation $(k = n)$ then MS decoder performance resembles that of MLSE [15].

## 4.2 Employing particular codes

### 4.2.1 Using a (3,1,2) convolutional encoder

In section 3.4 we described how MS decoder can be used without expanding the signal bandwidth. Scheme 4 of Fig. 3.4-12 was used as an example and comparable results versus the standard case of scheme 3 were displayed. A small disadvantage of scheme 4 is that MS decoding of convolutional encoder of Fig. 3.4-17 requires more calculations than ML detection of scheme 3. As will be showed in the next section, the number of stored paths and metrics per branch is $2^{k(C-1)} = 16$ times greater than that of the MLSE. It is interesting to see what happens if we employ another convolutional encoder instead, such as that of Fig. 4.2-1. The reason is that with a smaller constraint length $(C = 3)$, the computational encumbrance is reduced (4 times greater than the MLSE). Substitution of one encoder from the other results in formation of scheme 5 (Fig. 4.2-2).

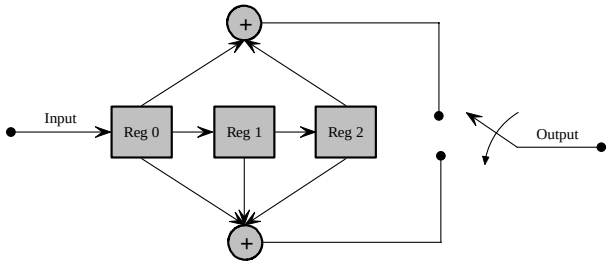


**FIGURE 4.2-1**
$C = 3,\ k = 1,\ n = 2$ convolutional encoder.

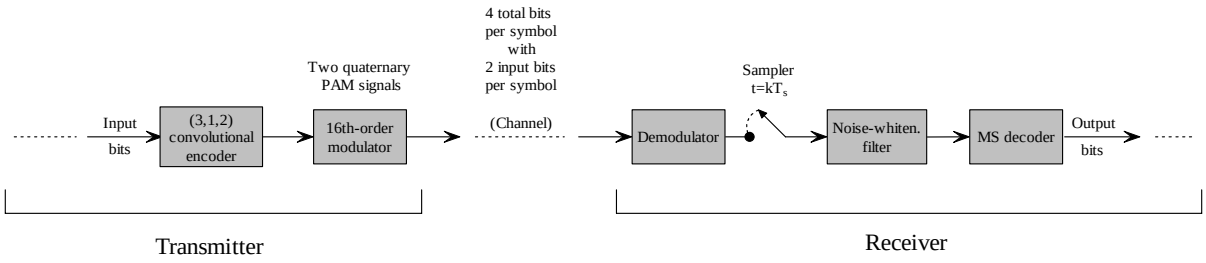


**FIGURE 4.2-2**
Employing MS decoder without increasing the bandwidth (scheme 5).

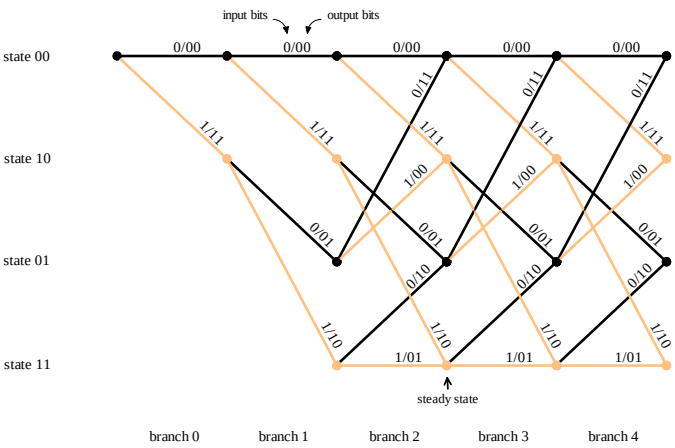


**FIGURE 4.2-3**
Trellis diagram for $C = 3,\ k = 1,\ n = 2$ convolutional encoder of Fig. 4.2-1.

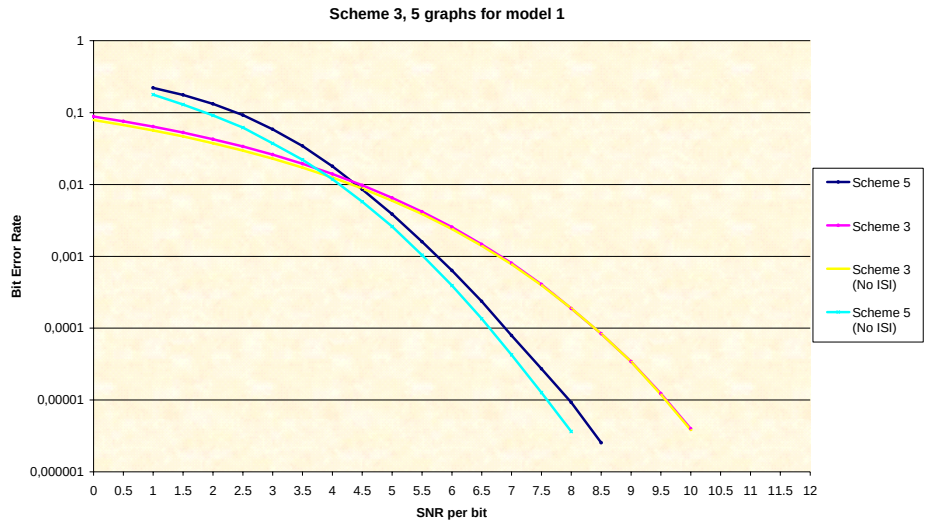


**FIGURE 4.2-4**
Scheme 3 and 5 performance for eq. discrete model 1.

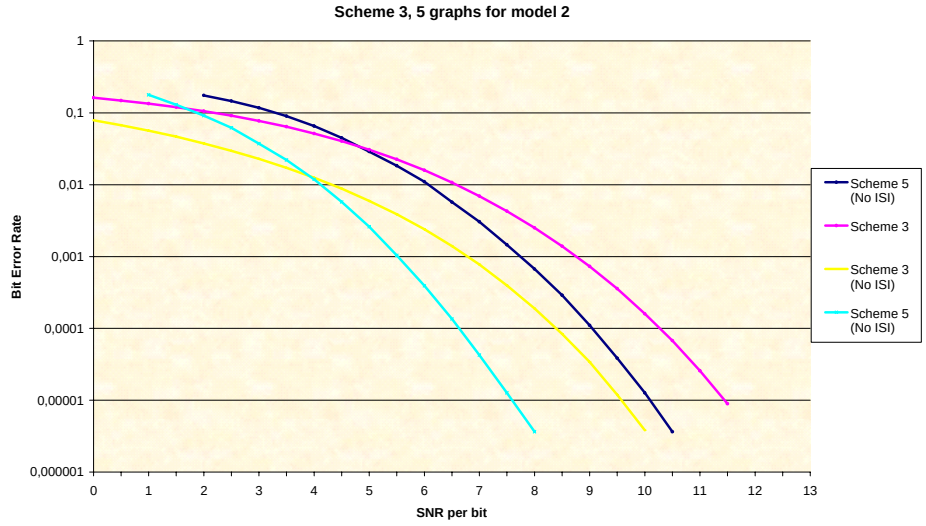


**FIGURE 4.2-5**
Scheme 3 and 5 performance for eq. discrete model 2.

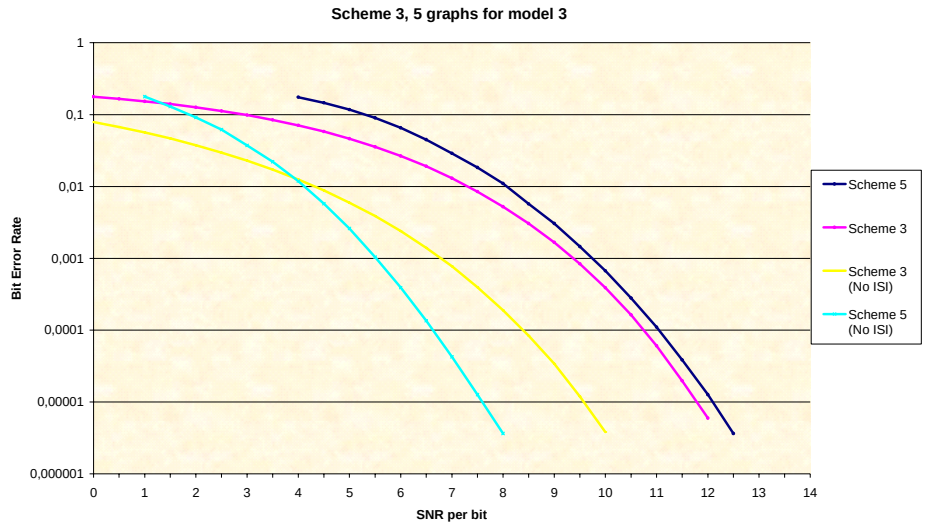


**FIGURE 4.2-6**
Scheme 3 and 5 performance for eq. discrete model 3.

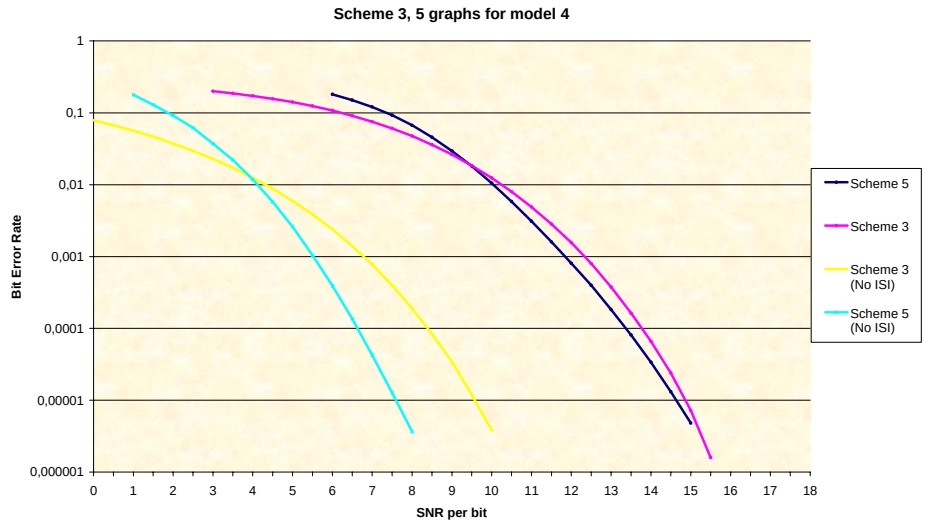


**FIGURE 4.2-7**
Scheme 3 and 5 performance for eq. discrete model 4.

We will focus on the performance near a $10^{-5}$ bit error rate. Concerning the first two models, we see that reductions in $E_b / N_0$ are accomplished in comparison with the standard scheme 3 scenario. However, the reductions are approximately $1\text{dB}$ smaller than the reductions observed for scheme 4 in Fig. 3.4-13 and 3.4-14. This is due to the smaller constraint length of the convolutional encoder used. Smaller constraint length implies fewer Trellis states and this results in decreased variety in total possible Trellis paths and lessening of the error correction properties. Concerning the third model, we notice that the performance of scheme 5 is $0.3\text{dB}$ worse than that of scheme 3. This is not as surprising as it seems. It happens sometimes when the two following conditions are met: An equivalent discrete model introducing very serious ISI with only one or two interfering coefficients and a convolutional encoder with small constraint length. Usually, channels that cause significant ISI are described by an equivalent discrete model response containing more than two interfering taps.

Therefore, model 3 is the only case where no improvement is achieved. As for the last model, we see that scheme 5 succeeds in reducing ISI in the critical areas, by $0.2$ to $0.4$ dB.

We note that in this case, the more ISI exists, the less power reduction can be accomplished. Nevertheless, even a $0.1$ dB reduction is considerable. In practical applications where diversity[1] techniques are used the absolute dB values of all of the above graphs are smaller.

Another conclusion extracted from the simulations is that the more (interfering) coefficients we have, the bigger improvement MS decoder can manage. This can be easily realised from the difference in the results between Fig. 4.2-6 and 4.2-7. Although channel model 4 introduces more ISI than model 3, it is "handled" in a more efficient way by MS decoder because of its larger ISI dispersion. A different helpful action would be to use a convolutional encoder with high constraint length. However, this strategy has its toll on computational complexity for scheme 5. It should be mentioned that the particular encoder of Fig. 4.2-1 is the best possible $(3,1,2)$ convolutional encoder that can be employed. It exhibits a $d_{FD} = 5$ (Fig. 4.2-3).

### 4.2.2 Using a (3,1,2) convolutional encoder & more bandwidth

In Fig. 4.2-4 to Fig. 4.2-7 we observed that in spite of employing higher modulation, power requirements can be reduced for bit error rates around $10^{-5}$ compared to the standard case of scheme 3. With the exception of model 3, a decrease in $E_b / N_0$ was accomplished for the other models through the use of scheme 5. In order to achieve an even greater reduction in SNR per bit, we perform simulations for scheme 8 displayed in Fig. 4.2-8. It contains the same devices with scheme 5, besides modulation. A lower order ($4$- th order) modulator is used. Using lower modulation results in transmitting one input bit per symbol instead of two (scheme 3). Consequently, we have to double the symbol rate in order to maintain the same information rate. Doubling the symbol rate means that double bandwidth is needed for the transmission, introducing more ISI and worsening the results. The question is therefore if what we "gain" from the modulation decrease is more than what we "lose" from the bandwidth expansion. The results obtained are shown in Fig. 4.2-9 to Fig. 4.2-12.

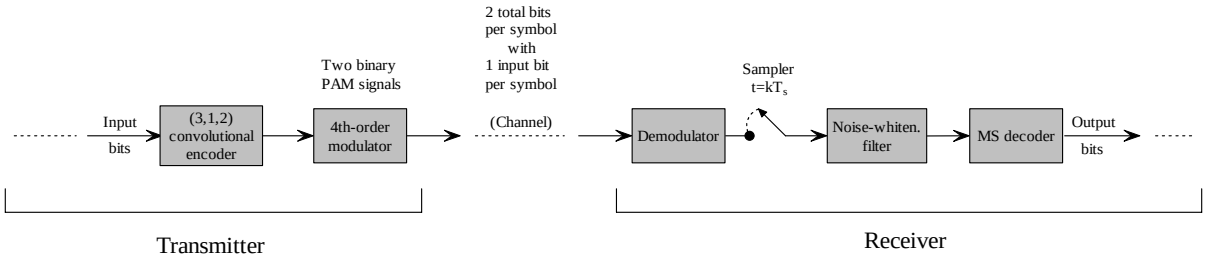


**FIGURE 4.2-8**
Employing MS decoder and increasing the bandwidth (scheme 8).

[1] Diversity is called the technique in which information is transmitted in different replicas. Resources for these replicas can be of many kinds, such as time slots, bandwidth, antennas, etc. The probability that a data segment is erroneous in all replicas is very small and thus quality of transmission and BER improves.

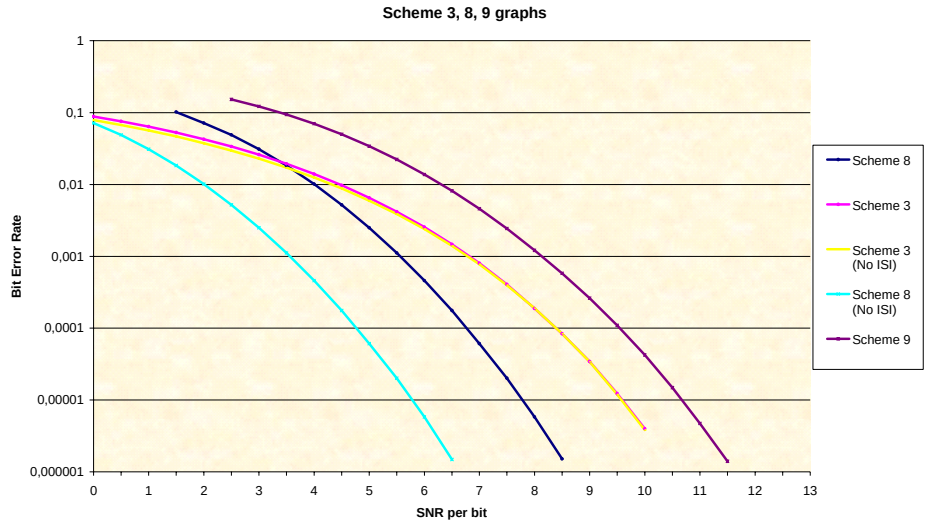


**FIGURE 4.2-9**
Scheme 8, 9 performance and scheme 3 performance for eq. discrete model 1.

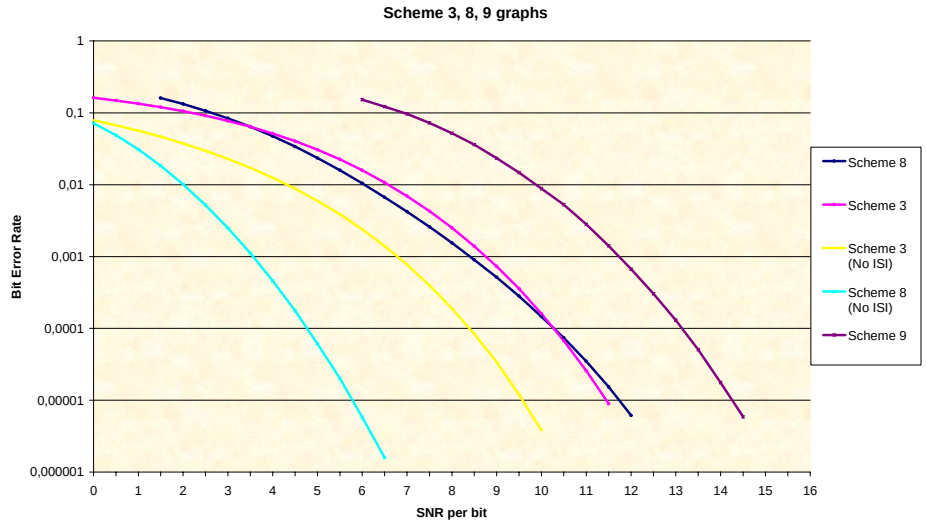


**FIGURE 4.2-10**
Scheme 8, 9 performance and scheme 3 performance for eq. discrete model 2.

For comparison reasons, another scheme (Fig. 4.2-13) is pictured on the graphs. Scheme 9 is similar to scheme 8 with the exception that it uses conventional means of equalising and decoding instead of MS decoding. A first observation to make for Fig. 4.2-9 to 4.2-12 has to with the large difference between the curves of scheme 8 and scheme 9. For a $10^{-5}$ bit error rate the difference fluctuates between $2.6\,\text{dB}$ (model 2) and $3.8\,\text{dB}$ (model 4). It is apparent that MS decoding is considerably beneficial, however not as much as in Fig. 3.4-7 to 3.4-10 in section 3.4. The reason is that here lower modulation is used. With lower modulation schemes, MLSE and Viterbi decoder can manage errors much better than with very high signalling such as $8$-PAM.

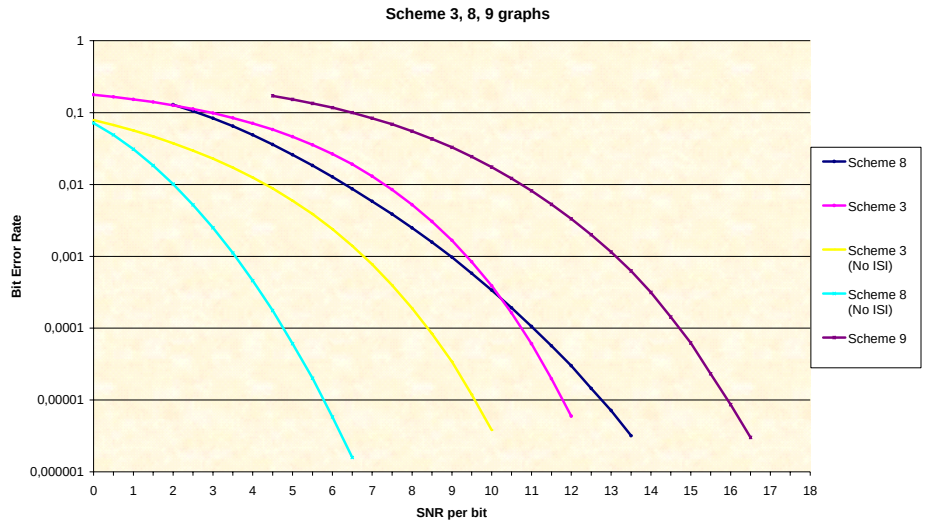


**FIGURE 4.2-11**
Scheme 8, 9 performance and scheme 3 performance for eq. discrete model 3.

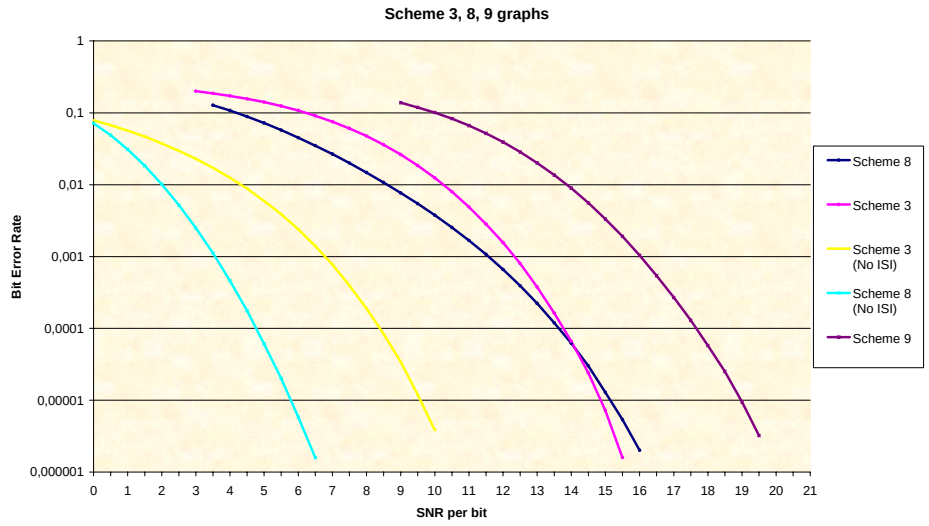


**FIGURE 4.2-12**
Scheme 8, 9 performance and scheme 3 performance for eq. discrete model 4.

We note that in the majority of cases scheme 8 does not yield an improvement versus scheme 3. For lower than $10^{-5}$ error levels, there is an SNR reduction only in good quality channels (model 1). For the rest of the models increasing the bandwidth proves disastrous, since excessive ISI is created. This is verified by the big distance between the scheme 8 graph and the scheme 8 (No ISI) graph. For model 4 the distance extends to an impressive $9\,\mathrm{dB}$ margin. Nevertheless, scheme 8 can be more useful as will be shown in the next chapter. Used as an internal part of an outer scheme that will include more coding, it can produce more competitive results in spite of the bandwidth expansion.

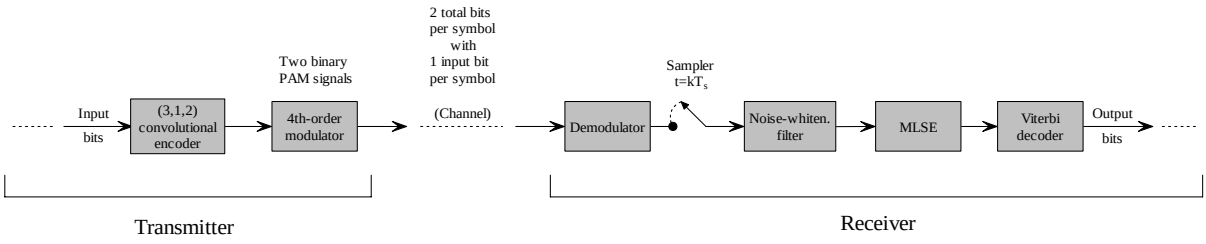


**FIGURE 4.2-13**
Standard scenario with MLSE and Viterbi decoding in place of MS decoding (scheme 9).

### 4.2.3 Using an encoder with TCM

Fig. 4.2-14 displays another potential use of MS decoder. It can be used in order to substitute devices of Fig. 4.2-15. Scheme 6 employs even higher modulation than scheme 7. It also employs the encoder of Fig. 3.4-18. Decoding is performed by MS decoder instead of MLSE. Preferably, a matched filter carries out demodulation. As always, this does not play an important role. Any kind of demodulator may be used, as long as the same demodulator is used for both comparable cases. Utilised bandwidth is the same for both schemes 6 and 7. Four input bits per symbol are transferred across and the same equivalent discrete model is used in both occasions. Simulations were conducted for models 1 and 3. The results can be seen in Fig. 4.2-16 and 4.2-17.

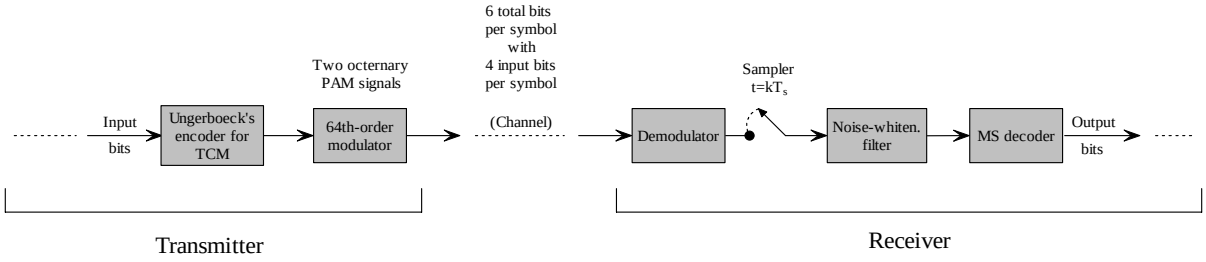


**FIGURE 4.2-14**
MS decoding for Ungerboeck's encoder (scheme 6).

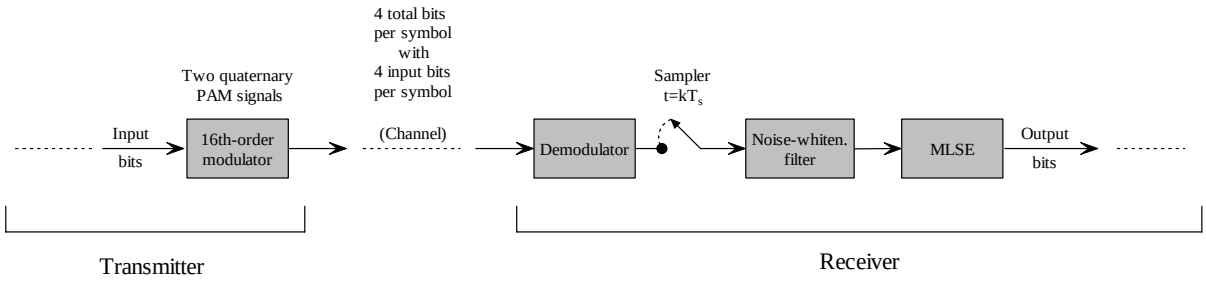


**FIGURE 4.2-15**
Conventional use of an ML detector (scheme 7).

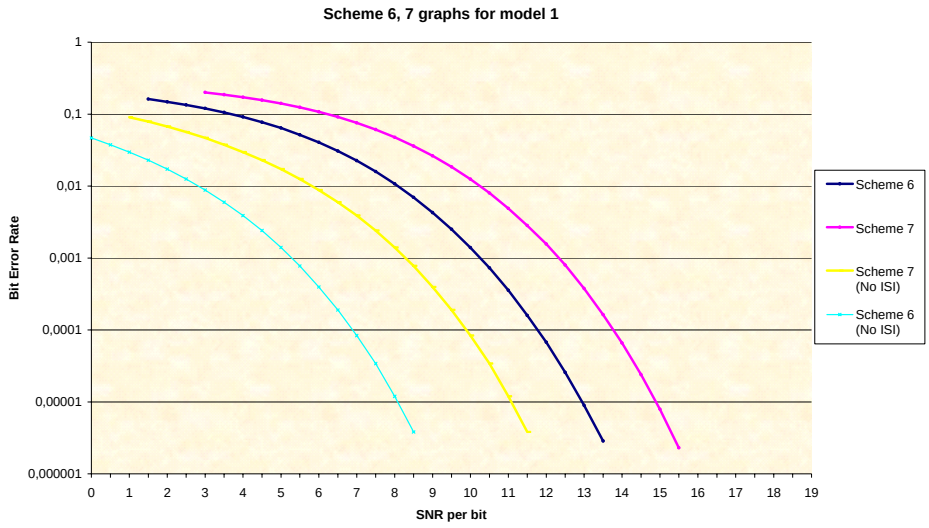


**FIGURE 4.2-16**
Scheme 6 and 7 performance for eq. discrete model 1.

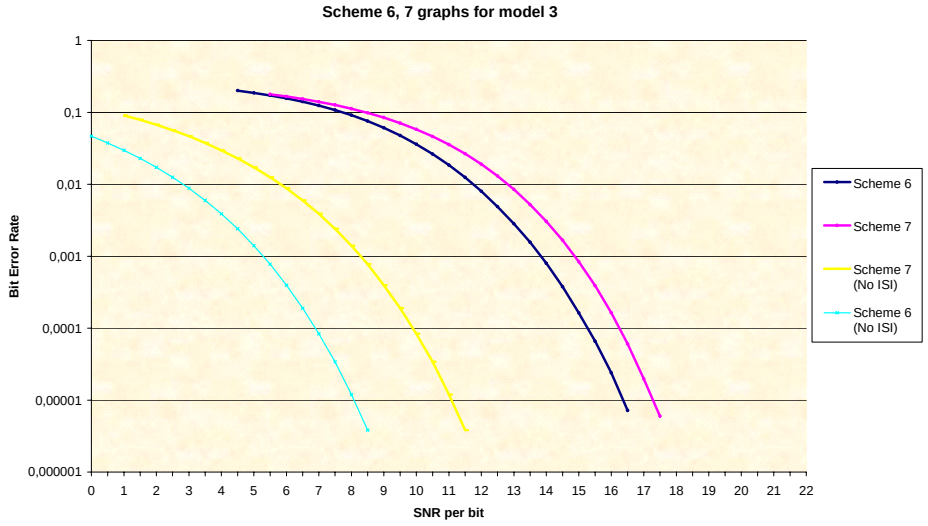


**FIGURE 4.2-17**
Scheme 6 and 7 performance for eq. discrete model 3.

Although scheme 6 uses higher modulation, it succeeds in reducing the SNR requirements. The reduction reaches around 2 dB for model 1 and 0.9 dB for model 3. The satisfactory results have mainly to do with three things. MS decoder employment, the specific encoder proposed by Ungerboeck and proper symbol mapping. Mapping was done according to Fig. 3.4-20. Scheme 6 is a typical case which shows that MS decoder can be employed beneficially with high modulation schemes as well.

## 4.3 Computational complexity

Employing optimum detection in a transmission, results in computational burden that sometimes cannot be affordable in practice. This happens because of the very large complexity introduced by MLSE or MAP equaliser. MS decoder that combines equalising and decoding, yields decoding results with a lower suffered complexity. In this section, we investigate the range of this improvement so as to extract useful conclusions. Examples are going to be used as reference.

In order to represent computational complexity of either Viterbi or MS decoding algorithm in a quick way, we will use the notation

$$Complexity \triangleq \{Cl, St\}$$

where $Cl$ is the average number of computed metrics when moving one branch forward through the Trellis,

$St$ is the average number of stored paths along with their metrics each time we move one branch forward through the Trellis.

MLSE operates with the use of Viterbi algorithm (section 2.2). As a result, the above notation can be used to represent the complexity of an ML detector as well. We consider three cases.

### 4.3.1 Basic use

In the basic use described in section 3.4, we have to do with two scenarios. The optimum standard technique, which employs ML detection and Viterbi decoding, and the MS decoding technique. In both cases $M$ - ary baseband signalling is used inside the modulator. In the first case, according to (3.2) we have

$$St_1^a = M^{(L+F)} \tag{4.29}$$

and according to (3.4)

$$St_1^b = 2^{k(C-1)} \tag{4.30}$$

where $a$ and $b$ denote the equalising (MLSE) and decoding Trellis respectively. On the other hand, for the second case

$$St_2 = 2^{k(C+B_L+B_F-1)} \tag{4.31}$$

We note that with proper selection of parameter $k$, $St_2$ is not greater than either $St_1^a$ or $St_1^b$. In fact, for large values of $M$, $St_2$ is significantly smaller. We verify that with the help of a few examples.

**1st example:**
We assume 8-level baseband PAM signalling, an equivalent discrete model with $L = F = 2,$ as well as a convolutional encoder of $C = 3$ and with a code rate $k/n = 1/3.$

MLSE computations require:

$\{Cl_1^a, St_1^a\} = \{32768,\ 4096\}$ as confirmed by (3.1) and (3.2)

Viterbi decoder computations require:

$\{Cl_1^b, St_1^b\} = \{8,\ 4\}$ as confirmed by (3.3) and (3.4)

MS decoder computations require:

$\{Cl_2, St_2\} = \{128,\ 64\}$ as confirmed by (3.6) and (3.7)

for carrying out equalising and decoding simultaneously.

These numbers represent the average complexity needed. In this example, 128 computations of metrics and 64 stored paths and metrics are required on average for a 1-bit MS decoder output.

**2nd example:**
16-level PAM, $L = 3,\ F = 2,\ C = 2,\ k = 2,\ n = 4.$

For MLSE:

$\{Cl_1^a, St_1^a\} = \{16.777216x10^6,\ 1.048576x10^6\}$

For Viterbi decoder:

$\{Cl_1^b, St_1^b\} = \{16,\ 4\}$

For MS decoder:

$\{Cl_2, St_2\} = \{16384,\ 4096\}$

**3rd example:**
Binary PAM, $L = 6,\ F = 0,\ C = 2,\ k = 2,\ n = 3.$

For MLSE:

$\{Cl_1^a, St_1^a\} = \{128,\ 64\}$

For Viterbi decoder:

$\{Cl_1^b, St_1^b\} = \{16,\ 4\}$

For MS decoder:

$\{Cl_2, St_2\} = \{256,\ 64\}$

**4th example:**
Binary PAM, $L = 6,\ F = 0,\ C = 3,\ k = 1,\ n = 2.$

For MLSE:

$\{Cl_1^a, St_1^a\} = \{128,\ 64\}$

For Viterbi decoder:

$\{Cl_1^b, St_1^b\} = \{8,\ 4\}$

For MS decoder:

$$\{Cl_2, St_2\} = \{64, 32\}$$

**5th example:**
4-PAM, $L = 4,\ F = 4,\ C = 3,\ k = 1,\ n = 2.$

For MLSE:

$$\{Cl_1^a, St_1^a\} = \{262144, 65536\}$$

For Viterbi decoder:

$$\{Cl_1^b, St_1^b\} = \{8, 4\}$$

For MS decoder:

$$\{Cl_2, St_2\} = \{2048, 1024\}$$

As can be seen, there is appreciable reduction of complexity in high-order modulations where $M$ is large. On the contrary, if $M < 2^k$ computational burden can be increased. Such a case is the third example. To counter this, we must set the size of the input bits $(k)$ per shift of the convolutional encoder as small as possible. MS decoder provides then satisfactory results like in the fourth example. In general, in order to minimise MS decoding complexity, it is customary to use a convolutional encoder with $n = \log_2 M.$ We ensure that each PAM symbol corresponds exactly to one Trellis branch. In a way, we "utilise" interfering branches $B_L$ and $B_F$ in the best way, in order to contain as much interfering symbols as possible. Of course, no considerations can be made regarding $L$ and $F$, since these parameters change from time to time. It is of no importance which between $L$ and $F$ is bigger, as far as complexity is concerned.

### 4.3.2 Universal use without bandwidth expansion

We have to do with two different modulations. Let $M_1$ be the level of the baseband modulation employed in the standard technique and $M_2$ the level of the baseband modulation employed in the MS technique. We focus on the case where $M_2 = M_1^{n/k}.$ In order to ensure this, we choose a convolutional encoder such that $k = \log_2 M_1.$ For the standard technique we have

$$St_1^a = 2^{k(L+F)} \tag{4.32}$$

where $a$ denotes the optimum equalising Trellis. Similar with previously, we have for the MS technique

$$St_2 = 2^{k(C+B_L+B_F-1)} \tag{4.33}$$

with

$$B_L = \left\lceil \frac{\log_2 M_2}{n} L \right\rceil,\ B_F = \left\lceil \frac{\log_2 M_2}{n} F \right\rceil$$

Since $k = \log_2 M_1 \Rightarrow M_2 = 2^n$ and hence $B_L$, $B_F$ become equal to $L$, $F$ respectively. Consequently (4.33) yields

$$St_2 = 2^{k(L+F)} 2^{k(C-1)} \tag{4.34}$$

Considering (4.32) and (4.34), we note that storing of metrics and paths increases in this scenario. We see that in this case, MS decoder complexity is $2^{k(C-1)}$ times greater than the complexity of the MLSE. The important thing is that the additional burden does not depend on the channel characteristics, since it is independent of parameters $L$, $F$. In general, MS technique showed in Fig. 3.4-12 increases complexity in a controlled manner. This increase has to with the last of the contained devices (MS decoder) and the convolutional encoder inserted before modulation.

Concerning the average number of the computed metrics in both cases, a similar increase by $2^{k(C-1)}$ times can be easily figured:

$$Cl_1^a = 2^{k(L+F+1)} \tag{4.35}$$

$$Cl_2 = 2^{k(L+F)} 2^{kC} \tag{4.36}$$

**1st example:**
We consider the case of Fig. 3.4-11 and Fig. 3.4-12 $(C = 5,\ k = 1,\ n = 2)$ for equivalent discrete model 3 $(L = F = 1)$.

MLSE average complexity per output bit:

$\{Cl_1^a, St_1^a\} = \{8,\ 4\}$ computed by (4.35) and (4.32)

MS decoder average complexity per output bit:

$\{Cl_2, St_2\} = \{128,\ 64\}$ computed by (4.36) and (4.34)

**2nd example:**
This scenario concerns scheme 3 of Fig. 3.4-11 and scheme 5 of Fig. 4.2-2 $(C = 3,\ k = 1,\ n = 2)$ for equivalent discrete model 4 $(L = F = 2)$.

For MLSE:

$\{Cl_1^a, St_1^a\} = \{32,\ 16\}$

For MS decoder:

$\{Cl_2, St_2\} = \{128,\ 64\}$

As expected, computational burden increases by a factor of $2^{k(C-1)}$ in all cases.

### 4.3.3 Universal use with bandwidth expansion

We consider a third MS decoder employment category. Some related performance graphs have been displayed in Fig. 4.2-9 to 4.2-12 (scheme 8) while more will be displayed in the next chapter. We present the category in general and analyse its complexity behaviour. Fig. 4.3-1 shows a scheme that includes devices at the end of a transmitter and at the start of a receiver. It is a generalisation of scheme in Fig. 3.4-11 and it is found in almost every telecommunication scenario.

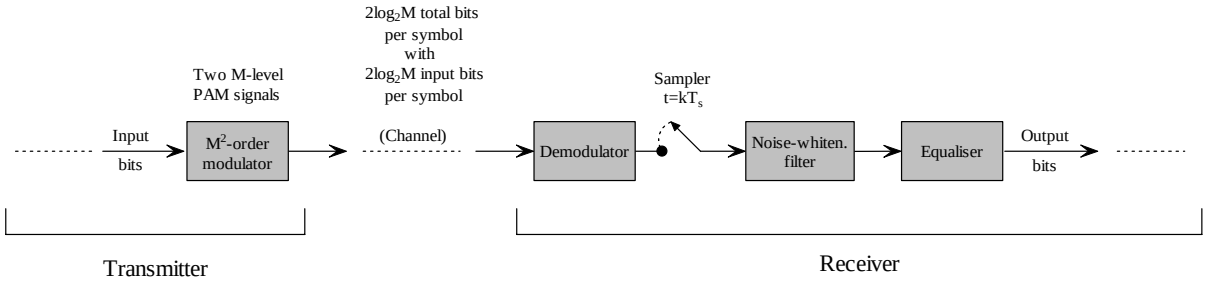


**FIGURE 4.3-1**
Standard technique.

The above devices can be substituted by those displayed in Fig. 4.3-2. What changes, is the insertion of a $k/n$ rate encoder before modulator and replacement of the equalising device by MS decoder. All other processes remain the same, including modulation. The only limitation that we have to take into account is to use an encoder with $n \geq \log_2 M$. If $n < \log_2 M$, one baseband symbol would not "fit" into one Trellis branch and that would be (as always) destructive. Insertion of the convolutional encoder causes an undesirable effect. $(k/n)2\log_2 M$ input bits are contained in each symbol instead of $2\log_2 M$ in the standard case. Since $k < n$ we understand that the information bitrate will be reduced. In order to counter this problem there are two possible solutions. The first is to increase modulation order while the second is to increase the symbol rate. The former is the case of universal use without bandwidth expansion described in section 3.4. The latter is displayed in Fig. 4.3-2. We are interested in such a scenario on the grounds that we can save on power at the cost of bandwidth. There exist applications where power is the main focus, more than bandwidth.

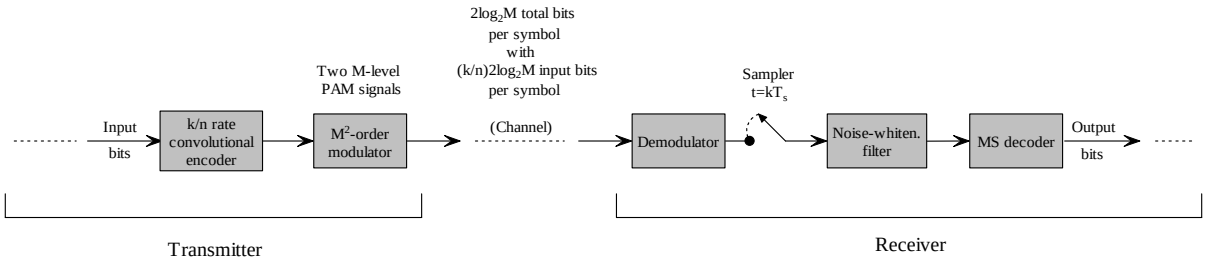


**FIGURE 4.3-2**
Employing MS decoder and increasing the bandwidth by $\lceil (n-k)/k \rceil 100\%$.

It is pointed out that $(k/n)2\log_2 M$ represents the average number of input bits per symbol. For example, for $\{k,n\} = \{2,3\}$ and $M = 2$ we have to do with a transmission involving $1.33$ input bits per symbol. This means that every three consecutive symbols contain four input bits and two redundancy bits. As always, terms "input" and "output" bits in Fig. 4.3-1 and Fig. 4.3-2 do not necessarily imply the input of the transmitter and the output of the receiver. Other devices may precede convolutional encoder in the transmitter or follow MS decoder in the receiver.

In order to estimate the complexity consequences of this strategy, we follow a procedure similar with previously. First of all, we notice that in both cases of Fig. 4.3-1 and Fig. 4.3-2 the same baseband modulation order is used, i.e. $M_1 = M_2 = M$. Hence, for the optimum equaliser (MLSE) of the standard case we have

$$\left\{Cl_1^a, St_1^a\right\} = \left\{M^{L+F+1}, M^{L+F}\right\} \tag{4.37}$$

and for MS decoder

$$\left\{Cl_2, St_2\right\} = \left\{2^{k(B_L+B_F)}2^{kC}, 2^{k(B_L+B_F)}2^{k(C-1)}\right\} \tag{4.38}$$

Next, we observe that $B_L \le L$ and $B_F \le F$ since $\log_2 M \le n$. Consequently, we infer:

$$k\left(B_L + B_F\right) \le k\left(L + F\right) \tag{4.39}$$

We want to estimate $St_2$ in comparison with $St_1^a$. We distinguish two cases:

- $k \le \log_2 M$. In other words, the input bits of the encoder we insert are equal to or fewer than the binary logarithm of the level of the PAM signalling used. In this case we have:

$$k\left(L + F\right) \le \left(L + F\right)\log_2 M \tag{4.40}$$

  With the help of (4.39) we gradually deduce:

$$\begin{gathered} k\left(B_L + B_F\right) \le \left(L + F\right)\log_2 M \\ k\left(B_L + B_F\right) \le \log_2 M^{L+F} \\ 2^{k(B_L+B_F)} \le M^{L+F} \end{gathered}$$

  Therefore we are able to set an upper boundary:

$$St_2 \le St_1^a 2^{k(C-1)} \tag{4.41}$$

  (4.41) shows that, independently of the channel, resulting complexity does not exceed $2^{k(C-1)}$ times the complexity of the standard case. In fact, with a proper insertion of an encoder having $k < \log_2 M$ we can sometimes accomplish significant complexity reduction instead of aggravation.

- $k > \log_2 M$. We know that $St_2$ is proportional to $St_1^{\alpha}$. We search for the values $\Sigma \in \Re$ such that

$$St_2 = \Sigma \cdot St_1^a \tag{4.42}$$

  Substitution of (4.37) and (4.38) yields

$$\begin{gathered} k\left(C + B_L + B_F - 1\right) = \log_2 M^{L+F} + \log_2 \Sigma \\ \Sigma = 2^{k(B_L+B_F)-(L+F)\log_2 M + k(C-1)} \end{gathered} \tag{4.43}$$

  From (3.5) we understand that:

$$B_L \in \left[\frac{L\log_2 M}{n}, \frac{L\log_2 M}{n} + 1\right) \tag{4.44}$$

  Similarly for $B_F$ :

$$B_F \in \left[ \frac{F \log_2 M}{n}, \frac{F \log_2 M}{n} + 1 \right) \tag{4.45}$$

Substituting the minimum and maximum values of $B_L$, $B_F$ in (4.43), we get the minimum and maximum value of $\Sigma$ respectively:

$$2^{\alpha(L+F)\log_2 M} 2^{k(C-1)} \leq \Sigma < 2^{\alpha(L+F)\log_2 M} 2^{k(C+1)} \tag{4.46}$$

where $\alpha = \frac{k}{n} - 1 < 0.$ We note that $\Sigma$ is not in any case greater than $2^{k(C+1)}$. Hence, the maximum value of any additional computational burden introduced by MS decoder, is irrelevant to the channel characteristics. However, channel response plays a role as far as reduction of computational complexity is concerned. As can be seen from (4.46), the bigger $L, F$ are, the greater reduction is achieved. This reduction is correct theoretically, however compensated in practice because greater bandwidth introduces more ISI. In the cases where $L, F$ are zero or close to zero, resulting complexity usually increases, but does not exceed $2^{k(C+1)}$ times the MLSE complexity of the standard case scenario.

Summarising, any increase in computational complexity can be bounded for this third strategy, independently of the channel characteristics. The value of the boundary depends exclusively on the parameters of the inserted convolutional encoder. What is important is that in the majority of cases complexity is not only kept within predefined limits but also decreases.

**1st example:**
We assume binary PAM signalling, an equivalent discrete model with $L = F = 2,$ as well as a convolutional encoder of $C = 3$ and with a code rate $k / n = 1/2.$

MLSE computations require:

$\{Cl_1^a, St_1^a\} = \{32, 16\}$ as found from (4.37)

MS decoder computations require:

$\{Cl_2, St_2\} = \{32, 16\}$ as found from (4.38)

**2nd example:**
Binary PAM, $L = 4,\ F = 2,\ C = 3,\ k = 1,\ n = 2.$

For MLSE:

$\{Cl_1^a, St_1^a\} = \{128, 64\}$

For MS decoder:

$\{Cl_2, St_2\} = \{64, 32\}$

**3rd example:**
Binary PAM, $L = 3,\ F = 3,\ C = 3,\ k = 1,\ n = 2.$

For MLSE:

$$\{Cl_1^a, St_1^a\} = \{128,\ 64\}$$

For MS decoder:

$$\{Cl_2, St_2\} = \{128,\ 64\}$$

**4th example:**
4-PAM, $L = 4,\ F = 4,\ C = 3,\ k = 1,\ n = 2.$

For MLSE:

$$\{Cl_1^a, St_1^a\} = \{262144,\ 65536\}$$

For MS decoder:

$$\{Cl_2, St_2\} = \{2048, 1024\}$$

In conclusion, using MS decoder appropriately can lead to machine relief. In this section, the complexities of the three main use categories were estimated. In every occasion of the first basic category as well as in many cases of the third category, the required number of calculations and storage space was reduced. As for the second category, we found out that computational complexity increases, but up to a predefined limit, independent of the channel characteristics.

## 4.4 Interleaving

The ordinary receiving procedure in wireless applications is an equaliser followed by a deinterleaver and that, in its turn, followed by a decoder. In all cases presented so far, we saw that the prerequisite in order to use MS decoding is to employ at least one convolutional encoder before modulating the transmitted signal. An important issue to examine is if it would be possible to interpolate an interlever between the encoder and modulator, i.e. modify MS decoding so that deinterleaving is also somehow performed internally.

Let us consider an approach with the help of an example. We assume a random transmission of a symbols sequence $s_k$ and the corresponding sequence of received values $y_k$, $k \in \aleph$. Without loss of generality, we assume an equivalent discrete model introducing four interfering symbols, two following and two preceding. Upon receiving groups of symbols

$$\ldots, y_2, y_3, y_4, y_5, y_6, y_7, y_8, y_9, y_{10}, y_{11}, \ldots$$

MS decoder tries to find the most probable path by taking into account five symbols at a time; the current one, two previous and two following. For instance, in the processing of symbol $y_6$ a surviving path is not influenced by a choice on symbol $s_2$ (corresponding to $y_2$) since the most probable transmitted value $s_2$ has been detected during processing of previous symbols. This happens of course because $s_2$ does not interfere to $s_6$ and hence, reception of $y_6$ (or any other symbols thereafter) gives no more information about the potential value of $s_2$.

On the other hand, if a time interleaver was employed between encoding and modulation, the received symbol sequence would be scrambled:

$$\ldots, y_{339}, y_{10}, y_{379}, y_{13}, y_{419}, y_{15}, y_{459}, y_{17}, y_{499}, y_{20}, \ldots$$

The numbers in the above sequence indicate the order at the output of the convolutional encoder. Since $F = 2$ and $L = 2$ MS decoder "is aware" that the value of a symbol has been corrupted by four of its neighbours. E.g. $s_{379}$ has been influenced because of ISI from the values of $s_{339}$, $s_{10}$, $s_{13}$ and $s_{419}$. Similarly, the value of $s_{13}$ has been influenced from the values of $s_{10}$, $s_{379}$, $s_{419}$ and $s_{15}$ and so on for all other symbols. All this has to do with equalising. As far as decoding is concerned, MS decoder knows that each symbol is dependable on the previous symbols as specified by Trellis diagram. At this point we have two cases:

i) Before all of $y_{12}, y_{11}, \ldots, y_0$ have been received, an estimation of the value of $s_{13}$ based on decoding knowledge (Trellis), cannot be made. The only estimation of $s_{13}$ that can be made, is based on ISI estimation, i.e. because of knowledge of $y_{10}$, $y_{379}$, $y_{419}$, $y_{15}$ and of course $y_{13}$. We come to the conclusion that only equalisation procedure can take place. For this procedure there is no limitation concerning the possible values of $s_{10}$, $s_{379}$, $s_{419}$, $s_{15}$ and $s_{13}$ which can be assigned any value from a total of $M$ values, where $M$ is the size of the PAM-symbol alphabet. Consequently, the computational burden of ML detection will have to be encountered and this yields no improvement.

ii) After all of $y_{12}, y_{11}, \ldots, y_0$ have been received, MS decoder is in position to compute the metrics of all paths containing the first 14 symbols. However, there still exist interfering symbols, over the possible values of which no limitation can be taken into

account, even with the help of the Trellis. For example, when considering paths ending in symbol $s_{13}$, such a following sum may have to be calculated:

$$\begin{aligned}&\ldots-\left[y_{11}-\left(f_{-2}s_{21}+f_{-1}s_{229}+f_0 s_{11}+f_1 s_{189}+f_2 s_8\right)\right]^2\\&-\left[y_{12}-\left(f_{-2}s_{14}+f_{-1}s_{139}+f_0 s_{12}+f_1 s_{99}+f_2 s_9\right)\right]^2\\&-\left[y_{13}-\left(f_{-2}s_{15}+f_{-1}s_{419}+f_0 s_{13}+f_1 s_{379}+f_2 s_{10}\right)\right]^2\end{aligned}$$

Although knowledge of channel encoding process limits the number of possible values for $s_8$, $s_9$, $s_{10}$, $s_{11}$, $s_{12}$, $s_{13}$, $s_{14}$, and $s_{15}$, there is no limitation for the values of $s_{21}$, $s_{99}$, $s_{139}$, $s_{189}$, $s_{229}$, $s_{379}$, and $s_{419}$ which can be assigned any value from a total of $M$ values. With the help of the Trellis, there can only be limitation for these values, if MS decoder considers all possible combinations of following branches in the Trellis that include every symbol until the last needed ($s_{419}$). But then, a great number of paths would have to be stored and this increases enormously, rather than decreases, computational complexity comparing to MLSE. Furthermore, one can easily understand the burden suffered if interleaver depth is very large.

Summarising, there can be no interleaving exactly before modulation, if an MS decoder is to be used. Interleaving eliminates the limitation on the number of possible combinations of transmitted symbols, accomplished by channel coding. Without the existence of this limitation, MS decoder yields no improvement not only in computational time but in result efficiency as well. Inability to use interleaving is quite catastrophic for wireless communications. However, alternative schemes can be devised in order to take advantage of MS decoding. Examples are the two types of universal use described in sections 3.4 and 4.3. Relative performance results are to be shown in the next chapter.

# 5. INTEGRATION IN GSM

## 5.1 Designing a telecommunication system

### 5.1.1 Basic parameters

There are four main parameters to take into account when designing a telecommunication system. All other technical characteristics can be derived after the limits of these parameters have been set.

1. Information bit rate

It is the goal of the entire communication. Particular care should be taken when comparing systems that employ different types of encoders in the transmitter. The transmission speed of the bits at the input of the first of these encoders should be taken as a reference.

2. Bandwidth

Bandwidth depends linearly on the transmission symbol rate and not the bit rate. For baseband signalling, the minimum bandwidth required is equal to $f_s/2,$ where $f_s$ is the symbol rate. This is the Nyquist theorem. For bandlimited modulation, i.e. for all commercial systems, the minimum bandwidth is equal to $f_s$.

3. Power

When speaking of power, we actually refer to the energy wasted for the transmission needs. $E_b/N_0$ is most of the times preferred to $S/N$ (section 4.1). Both are ratios; however, use of normalised noise power $(N_0)$ makes the analysis independent of the bandwidth, i.e. independent of the symbol rate. This allows systems to be compared in a very generic manner. Additionally, use of energy per information bit (and not coded bit) $E_b$ makes the analysis independent of the coding employed. In this way, systems with codes with different rates can be compared at a glance.

4. Complexity

Represents the average computational burden suffered by the machine. There is no standard way of estimating it. For every algorithm of course, the order of the number of calculations can be set within limits, like in section 3.2. An objective way of comparing the encumbrance between different devices was presented in section 4.3. In general, a manufacturer tries to design as simple a standard as possible since complexity not only affects operation time but cost as well.

In order to conduct comparisons between different schemes, one basic assumption is necessary. We assume that the same information bit rate is achieved. An important issue arises when it comes to which modulation order to be used. If we increase the order, we increase the number of bits per symbol and consequently the symbol rate decreases. This means that we save considerably on bandwidth. With a first look, saving on bandwidth seems to relieve power. Indeed two factors contribute to this direction. First, we notice:

$$\frac{E_b}{N_0} = \frac{E_b B_n}{N} \qquad (5.1)$$

Therefore, lessening bandwidth $B_n$ reduces $E_b/N_0$ too. Secondly, we consider that the smaller the symbol rate is, the bigger the symbol period is set. We recall from section 1.4 that a bigger period

decreases ISI since successive symbols are more scarcely placed upon sampling. ISI depletion has always beneficial effect upon power.

In spite of all the above, power is not reduced. The reason can be seen in equation (4.5). An increase in $M$ enlarges average signal power $S$ and $E_b / N_0$ exponentially. The exponential enlargement is greater than any relief of an $M$ increment. As a result, increasing the modulation order enables us to save on bandwidth and waste more power. On the other hand, reducing the modulation order decreases power but necessitates more bandwidth. It all depends on where the manufacturer wants to emphasise.

Another interesting issue to examine is the effect of the insertion of an additional encoder. Let $k/n$ be the rate of this encoder. In order to maintain the same information bit rate, there are two options. Either to increase the modulation order, or to increase bandwidth. The first case was covered previously. The second case involves a bandwidth increase by $(n-k)k^{-1}100\%$. From (5.1) we see that $E_b / N_0$ increases, with a first look. We also consider that the symbol period is reduced and this results in worsening of the ISI phenomenon. However, there is an essential point to be made. Information in this case is more resilient against noise and interference because of the additional redundancy. The redundancy is of the order of $(n-k)k^{-1}100\%,$ equal to the bandwidth increase. We conclude that no concrete prognosis can be made, as far as power requirements are concerned. Power may increase or decrease depending on the case.

In order to avoid such reference problems that have to do with the utilised bandwidth, we take two measures. We normalise average noise power $(N)$ and information bit rate $(f_b)$ towards bandwidth. The first leads to the definition of $E_b / N_0,$ as already described. The second results in a parameter called *bandwidth efficiency* or *spectral efficiency*. It is defined as the ratio of information bit rate to bandwidth and expressed in $b/\sec/Hz$:

$$\eta = \frac{f_b}{B_n} \tag{5.2}$$

### 5.1.2 Optimum receiver prerequisites

An optimum receiver must satisfy two criteria, the Nyquist theorem and the Matched-filter theorem. The Nyquist theorem says that the minimum bandwidth of a signal (of $f_s$ symbol rate) is $f_s/2.$ It also says that the ideal frequency response of the convolution between transmitter filter, channel and receiver filter must be flat, as shown in (1.12). There are many practical choices for $X(f)$ in order to have a flat spectrum for (1.12), if of course the bandwidth is at least $f_s/2.$ The most popular choice is the raised cosine spectrum

$$X(f) = \begin{cases} T_s & \text{for} \quad 0 \le |f| \le \frac{1-\beta}{2T_s}, \\ \frac{T_s}{2}\left[1+\cos\frac{\pi T_s}{\beta}(|f|-\frac{1-\beta}{2T_s})\right] & \text{for} \quad \frac{1-\beta}{2T_s} \le |f| \le \frac{1+\beta}{2T_s}, \\ 0 & \text{for} \quad |f| > \frac{1+\beta}{2T_s}, \end{cases} \tag{5.3}$$

Its impulse response is given in (1.1). Compared with the $\sin c$ function, the main advantage is that the raised cosine decays faster (Fig. 5.1-1). The attenuation of $x(t)$ is proportional to $1/t^3,$ better than that of the $\sin c$ pulse that is proportional to $1/t.$ Hence, the raised cosine spectrum gives a pulse that is less sensitive to timing errors. Roll-off factor $\beta$ is always bigger than 0.19 in practice.

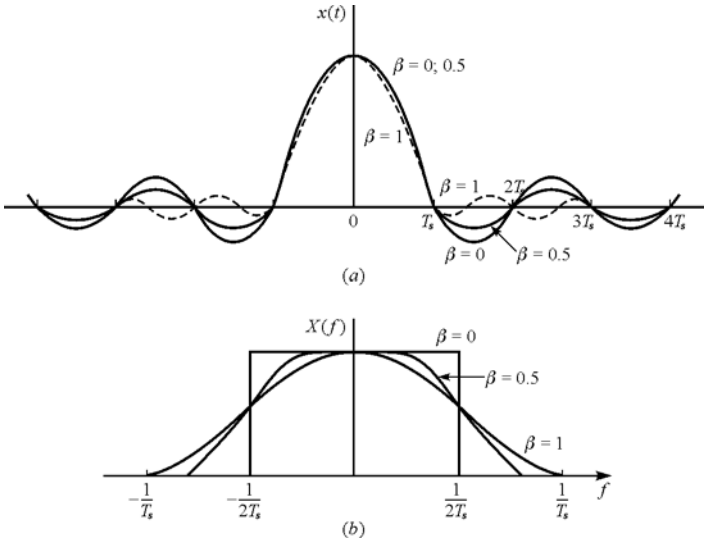


**FIGURE 5.1-1**
Pulses having a raised cosine spectrum.

Let $H(f)$ be the frequency response of the channel. We will first study what the manufacturer has to do to avoid ISI if the channel is ideal, i.e. if $H(f)=1.$ With the selection of a raised cosine shaped $X(f),$ we have

$$X(f)=P(f)P^*(f)=\left|P(f)\right|^2=\left|H(f)\Phi(f)\right|^2 \tag{5.4}$$

with all notations defined in section 1.4. Since we assumed that $H(f)=1,$ i.e. that the channel does not introduce any distortion within its passband, an obvious choice for the transmitter filter $\Phi(f)$ would be $\sqrt{X(f)}.$ Then $P(f)=\sqrt{X(f)}$ and the transfer function of the matched filter would also be

$$\frac{P^*(f)}{\|p(t)\|}=\frac{\sqrt{X(f)}}{\|p(t)\|} \tag{5.5}$$

Consequently, transmitter and receiver filter must be identical. This is referred to as the Matched-filter theorem. The overall optimum filtering strategy is that of a resulting raised-cosine spectrum, equally split among the transmitter and receiver filter. This statement is held for any baseband or modulated bandlimited system with any type of signalling or modulation.

We are now in position to estimate the actual transmission bandwidth. As can be "guessed" from Fig. 5.1-1b it is $f_s(1+\beta).$ We verify this easily with a transmission example of an $N$ - order PSK signal. The unfiltered power spectrum of the signal is equal to

$$S_{N-PSK}=KA^2T_s\left(\frac{\sin\pi(f-f_c)T_s}{\pi(f-f_c)T_s}\right)^2 \tag{5.6}$$

where $2A$ is the signalling distance two successive PAM symbols, $f_c$ is the carrier frequency and $K$ is a proportionality constant. After transmission and reception with square root raised-cosine filters the resulting filtered $N$ - PSK signal can be easily obtained as shown in Fig. 5.1-2.

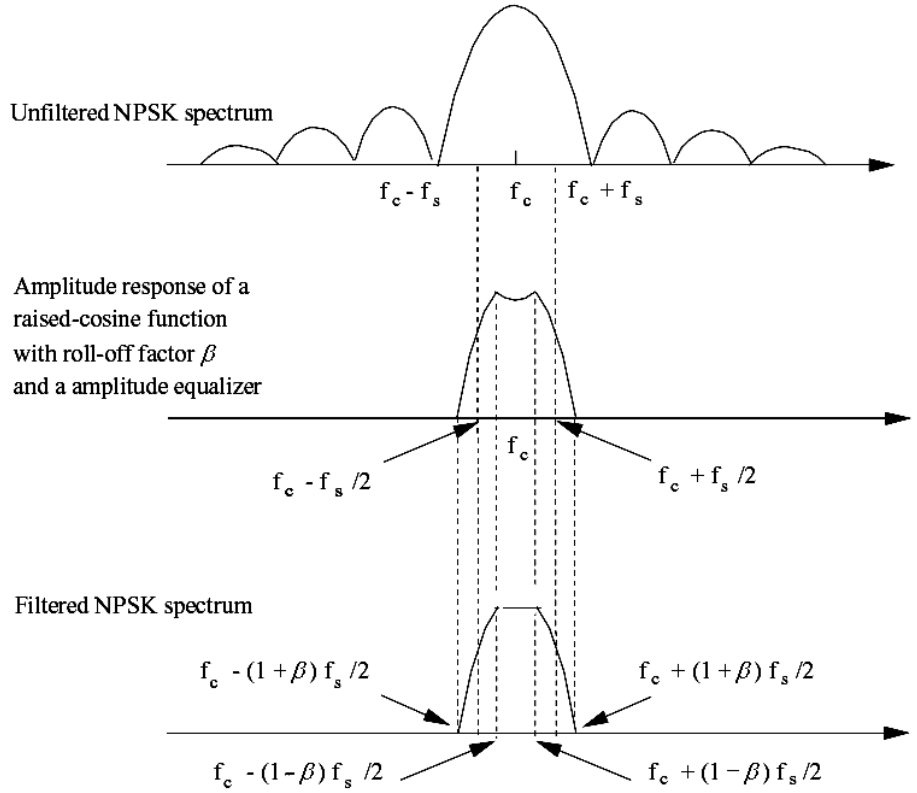


**FIGURE 5.1-2**
Power spectral density of a filtered $N$ - PSK signal.

We understand that the spectral efficiency of an $N$ - order modulated bandlimited transmission is

$$\eta = \frac{f_s \log_2 N}{f_s(1+\beta)} = \frac{\log_2 N}{1+\beta} \qquad (5.7)$$

The above filtering strategy theoretically creates an ISI-free mechanism. In practice, it is in general difficult to construct the appropriate analogue filters for the transmitter and receiver. The channel response $h(t)$ must be known in advance so that appropriate $P(f)$ and $P^*(f)/\|p(t)\|$ are designed. This is hardly practical, since in the majority of cases, $h(t)$ not only does introduce distortion and ISI $(H(f) \neq 1)$, but it is time-variant as well.

However, if we take into consideration that, what we want are the samples at intervals $T_s$ at the receiver, we may choose to build a simpler matched filter and put a digital filter, called equaliser, at the output to eliminate ISI. The main advantage of such an approach would have to do with the fact that a digital filter is easy to build and to alter for different countering-ISI schemes, as well as to fit for different channel conditions. This is the reason why equalisation is always necessary if we have to deal with ISI.

### 5.1.3 An example

We consider the design of a communication system required to establish $43.2$ Kbps transmission over a $50$ KHz channel bandwidth. We assume that encoding processes in the transmitter introduce $100\%$ redundancy, including ciphering or any other type of operation. Naturally, transmitter and receiver filters will follow the afore-mentioned tactic; however, what kind of modulation order is it better to be used?

Since the amount of data to be transmitted is doubled after the transmitter operations, we understand that we have to maintain a $2x43.2 = 86.4x10^3$ bit stream per second, as far as the coded bits are concerned. Hence, we have to achieve an efficiency of the order

$$\eta = 1.728 \ \ coded \ bits / \sec / Hz. \qquad (5.8)$$

Although design varies according to the application, the usual and profitable strategy in telecommunications is to employ as low modulation as possible. We try to find the minimum $N$ for which (5.8) is satisfied. Through (5.7), we see that for the values $N = 2,4$ (e.g. BPSK, QPSK) the maximum spectral efficiency achieved is $0.84$ and $1.68$ $b/\sec/Hz$ respectively. We have considered the best practical scenario with $\beta$ as low as $0.19.$ We realise that these choices do not meet (5.8) requirements. Therefore, we come to the conclusion that $8$-order modulation is better to be used, such as $8$- PSK.

## 5.2 GSM Full Rate Speech Channel (TCH/FS)

### 5.2.1 Physical layer specifications

In 1982 European Commission reserved frequencies around the 900MHz spectrum in all country members, setting the stage for interoperability across Europe [30]. From 1982 to 1990 the specifications of GSM were being decided upon and after two years the first operation network was available. Fig. 5.2-1 displays some general characteristics of GSM transmission.

| | ***GSM-900*** | ***GSM-1800*** |
|---|---|---|
| Downlink frequencies | 935-960 MHz | 1710-1785 MHz |
| Uplink frequencies | 890-915 MHz | 1805-1880 MHz |
| Channel spacing | 200 KHz | 200 KHz |
| Modulation | GMSK | GMSK |
| Typical MS transmit power | 3.7 mW to 1 W | 250 mW to 2 W |
| Maximum BS transmit power | 320 W | 20 W |
| Maximum distance per cell | 35 km | 8 km |
| Source (speech) encoding | LPC (13kbit) | LPC (13kbit) |

**FIGURE 5.2-1**
General specifications for GSM-900 and GSM-1800.

Downlink transmission takes place with the sending of data from the base station (BS) to the mobile (MS). Uplink transmission is considered the opposite. Because of diversity techniques, power requirements shown in Fig. 5.2-1 are significantly smaller than the ones displayed in the simulations of this thesis. There are various kinds of diversity that can be employed in practice. They result in improving transmission and lessening the required power. In this thesis, different standards including GSM were compared assuming a single-transmitter single-receiver mode, i.e. no diversity was used.

Fig. 5.2-2 displays the physical layer block diagram of a GSM transmitter. Source and channel coding are firstly carried out. Next, are interleaving and the necessary multiple access and encryption procedures. Multiple access techniques are used so that the same channel resources are shared by many users simultaneously. Finally, a $4$- th order modulator, namely Gaussian Minimum Shift Keying (GMSK) is used.

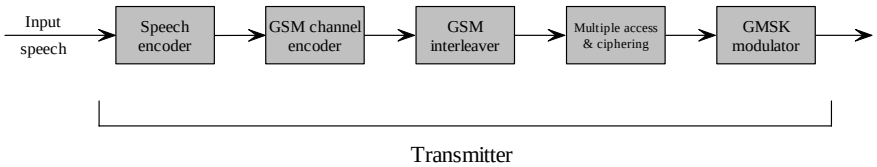


**FIGURE 5.2-2**
A GSM transmitter.

**RPE-LPC speech encoder**

An LPC encoder fits a given speech signal against a set of vocal characteristics. The best-fit parameters are transmitted and used by the decoder to generate synthetic speech that is similar to the original. Information from previous samples is used to predict the current sample. The coefficients of the linear combination of the previous samples, plus an encoded form of the *residual* represent the signal. Residual in source coding is the difference between the predicted and actual sample. Speech is divided into samples, giving an output bit rate of 13 Kbps.

**GSM channel encoder**

RPE-LPC Encoder produces a block of 260 bits every 20ms. It was found though testing that some of the 260 bits were more important when compared to others. As a result, they were categorised:

Class 1a – 50 bits (most sensitive to errors)
Class 1b – 132 bits (moderately sensitive to errors)
Class 2 – 78 bits (least sensitive to errors)

GSM adds redundancy bits to each of the three Classes differently. The Class 1a bits are encoded in a cyclic encoder. The Class 1b bits, together with the encoded Class 1a bits, are encoded using convolutional encoding. Finally, the Class 2 bits are merely added to the result of the convolutional encoder.

1. Cyclic encoding

The Class 1a bits are protected by three parity bits used for error detection. Cyclic codes are linear codes, i.e. the sum of any two codes is also a codeword. In addition to being linear, a cyclic rotation of a codeword produces another codeword. The code used in GSM is a $(53,50)$ code and thus, the generator polynomial used in the encoding is of degree $53-50=3.$ The specific polynomial used in GSM is $D^3+D+1$ as can be seen in Fig. 5.2-3. Once the data has been completely shifted through the system, the contents of the three registers will contain the three additional bits.

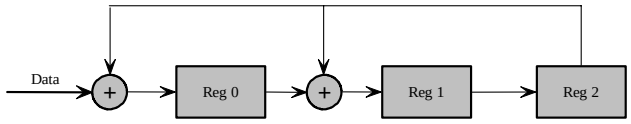


**FIGURE 5.2-3**
GSM cyclic encoder.

Cyclic encoding was chosen due to the ability to quickly determine if errors are present. The three redundancy bits produced by the cyclic encoder enable the receiver to quickly determine if an error was produced. If an error was produced the current 53-bit frame is discarded and replaced by the last known "good" frame [31].

2. Convolutional encoding

After the cyclic encoder, the 50 bits of Class 1a are added to the 132 Class 1b bits. The resulting 182 bits are reordered into an 185-bit sequence. The 91 bits in even positions take the first 91 positions (0-90) in the reordered sequence and the 91 bits in odd positions take the last 91 positions (94-184). Intermediate positions (91-93) are taken by the three parity bits produced by cyclic encoder. A tail of four extra '0' bits is added so that the encoder may be flushed. The total of 189 bits is encoded using the convolutional encoder of Fig. 5.2-4.

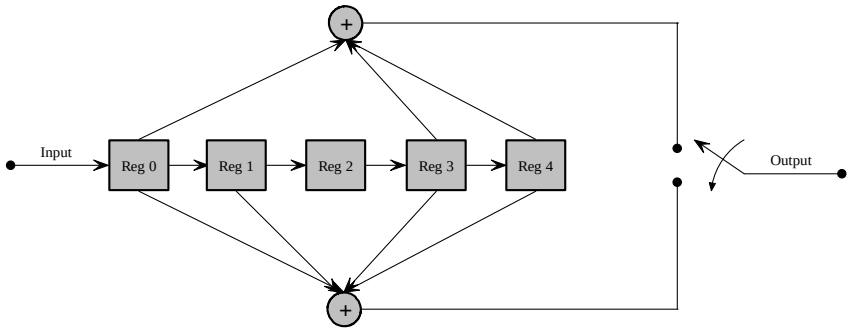


**FIGURE 5.2-4**
$C = 5,\ k = 1,\ n = 2$ convolutional encoder.

The reason for choosing the particular encoder has to do with its good error correction properties. Once the convolutional encoder has encoded the bits, a new sequence of 378 bits is produced. These 378 bits are directly added to the 78 Class 2 bits. As a result, the channel encoded bit sequence is now 456 bits long. Each 20ms burst produces 456 bits at a bit rate of 22.8 Kbps.

**Interleaving**

A form of block diagonal interleaving is used. If we denote with $k,\ k = 0,1,...,455$ the index of a bit in a delivered block (before interleaving) and with $A,\ A = 0,1,..., P, P+1,...\ P \in \aleph,$ the index of the delivered block, then the encoded bits are reordered and interleaved according to the following rule:

$$i(B, j) = e(A, k)$$

$$B = B_0 + 4A + (k \bmod 8)$$
$$j = 2\left[(49k) \bmod 57\right] + \left[(k \bmod 8) div 4\right]$$

where $e(A, k)$ are the encoded data bits
$i(B, j)$ are the interleaved data bits
$j$ is the index of bits in an interleaved data block,
$B$ is used for numbering of interleaved data blocks
$B_0$ marks the first block carrying bits from the data block with $A = 0$ (first data block in the transmission).

The result of the interleaving is a distribution of the reordered 456 bits of a given data block $A = P$ over eight blocks using the even numbered bits of the first four blocks $(B = B_0 + 4P + 0,1,2,3)$ and odd numbered bits of the last four blocks $(B = B_0 + 4P + 4,5,6,7)$. The reordered bits of the following data block $A = P + 1$ use the even numbered bits of the blocks $B = B_0 + 4P + 4,5,6,7$ $(B = B_0 + 4(P+1) + 0,1,2,3)$ and the odd numbered bits of the blocks $(B = B_0 + 4(P+1) + 4,5,6,7)$. Continuing with the next data blocks shows that one block always carries 57 bits of data from one data block $(A = P)$ and 57 bits of data from the next block $(A = P+1),$ where the bits from the data block with the higher number always are the even numbered data bits, and those of the data block with the lower number are the odd numbered bits.

**Multiple access**

The Multiple access Scheme defines how radio frequency can be shared by different simultaneous communication between different mobile stations located in different cells. A combination of Time-Division Multiple Access (TDMA) and Frequency-Division Multiple Access (FDMA) is used in order to share the limited bandwidth that is provided by regulators. FDMA divides the spectrum into small slices, and then each frequency slice is separated in time into eight *time slots* by TDMA. The set of eight time slots comprise a TDMA *frame*. In the standard GSM protocol, an individual user receives a pair of frequencies (one for downlink and one for uplink) and one time slot. The transmission of the voice signal is no longer continuous because of the time division, and hence, the data is transmitted in bursts. The assembly operation takes the final encoded and interleaved data and groups it into bursts. More about multiple access can be found in [32] and [33].

**Ciphering**

The two parties involved in encrypting and decrypting the data are the Authentication Center (AuC) and the SIM card in the mobile phone. Each SIM card holds a unique secret key, which is known by the AuC. The SIM card and AuC follow a few algorithms to first authenticate the user, and then encrypt and decrypt the data. For authentication, the AuC sends an 128-bit random number to the mobile phone. The SIM card uses its secret key and an algorithm called “A3”, to perform a function on the random number and sends back the 32-bit result. Since the AuC knows the SIM card's secret key, it performs the same function, and checks that the result obtained from the mobile phone matches its own result. If it does, the mobile user is authenticated.

Once authentication has been performed, the random number and the secret key are used in the “A8” algorithm to obtain a 64-bit ciphering key. This ciphering key is used with the TDMA frame number in the “A5” algorithm to generate a 114-bit sequence. It is important to note that the ciphering key is constant throughout a conversation, but the 114-bit sequence is different for every TDMA frame. The 114-bit sequence is processed by XOR circuits with the two 57-bit blocks in a TDMA burst [34]. The only user that can decrypt the data is the mobile phone or the AuC since they are the only ones that have access to the secret key, which is needed to generate the ciphering key, and the 114 bit sequence. None of the algorithms is known to the public; however some information about A5 has been leaked. It is known that A5 has a 40-bit key length, which allows for the encryption to be broken in a matter of days, but since cellular calls have a short lifetime, the weakness of the algorithm is not an issue. In the writer’s opinion, the only way to breach GSM privacy is through the construction or the imitation of a Base Station (BS). This is of course not practical, but not impossible either [35].

**GMSK modulator**

Most digital radio transmission systems today operate their high power amplifiers near saturation for maximum power efficiency. During this operating mode, the amplifiers introduce nonlinear amplitude and phase distortions. One of the damaging effects of these nonlinearities is the spectral spreading of the transmitted signal which increases undesirable interference to the adjacent channels.

Because of their large phase transitions (±90, 180 degrees) QPSK signals, when bandlimited, have significant envelope variations. As a result, when a bandlimited QPSK signal passes through a nonlinear amplifier operating at saturation, there is almost a total regeneration of the filtered frequencies (side-lobes) to their unfiltered levels (Fig. 5.2-5).

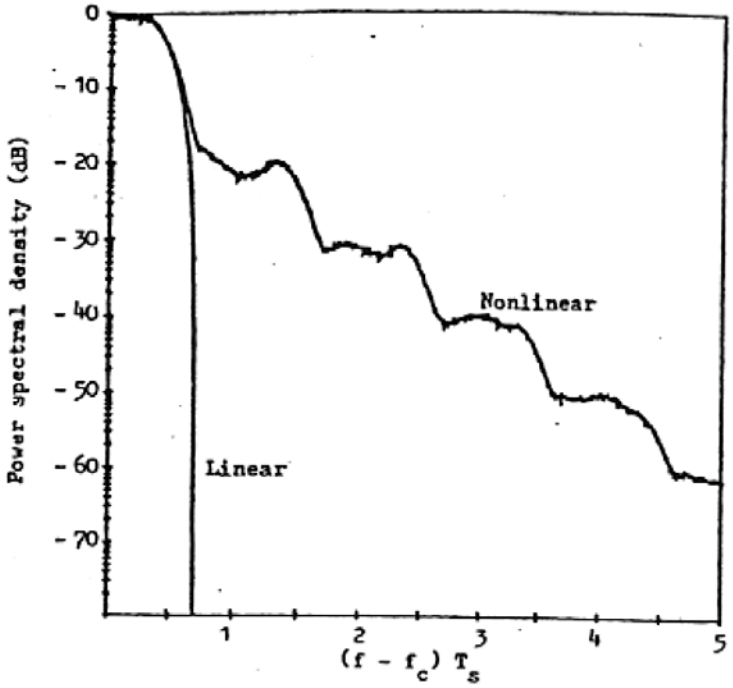


**FIGURE 5.2-5**
Spectrum spreading of a bandlimited QPSK signal at the output of an amplifier operating at saturation.

For nonlinear systems, alternative modulation schemes with reduced phase transitions are employed. Minimum Shift Keying (MSK) is such a method. When transmitted through a nonlinear amplifier, bandlimited MSK signals cause significantly less spectral spreading. This is the reason why GMSK is used in GSM.

An MSK signal can be expressed as

$$s(t) = \sqrt{\frac{2E_b}{T_b}} \cos[\omega_c t + \theta(t)] \qquad (5.9)$$

where

$$\theta(t) = \theta(t - T_b) + \frac{\pi}{2} \qquad \text{if a '1' bit was sent}$$

$$\theta(t) = \theta(t - T_b) - \frac{\pi}{2} \qquad \text{if a '0' bit was sent}$$

As can be seen, differential encoding is used. Current phase $\theta(t)$ depends not only on the current information bit, but on what was previously sent as well. The phase transitions for MSK are ±90 degrees, "smoother" than that of QPSK. For consecutive information bits of the same logic value, the absolute value of the phase does not go to infinity, but rotates around $0^\circ$ (Fig. 5.2-6).

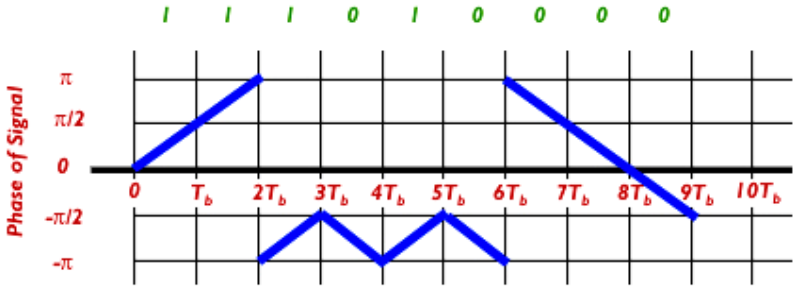


**FIGURE 5.2-6**
Phase fluctuation for an MSK signal.

Even though MSK power spectrum density falls quite fast, it does not fall fast enough so that interference between adjacent signals in the frequency band can be avoided. To take care of this, the original binary signal is passed through a Gaussian-shaped filter before it is modulated with MSK.

### 5.2.2 Considering conventional modifications

The usual strategy in telecommunications is to standardise the transmitter processes. The receiver devices are left in the discretion of the manufacturer, although recommendations are made. Most service providers employ MLSE for equalising and Viterbi decoder for convolutional decoding. In order to make comparisons versus GSM standard, we consider scheme 10 displayed in Fig. 5.2-7. Multiple access has been omitted for reasons of convenience. For understandable reasons, ciphering can also not be included.

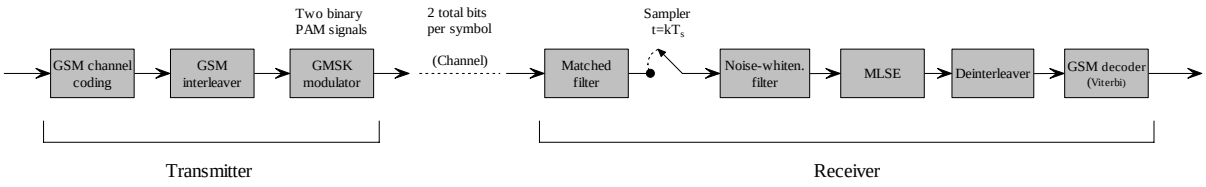


**FIGURE 5.2-7**
GSM standard (scheme 10).

Reasonably, GSM specifications were chosen in order to ensure minimal power consumption. The use of binary level PAM is a clear indication concerning this issue. Any baseband signalling of higher level would save on bandwidth but would demand more power. That would be undesirable for health reasons. We therefore focus on if there exists a way to further decrease power requirements. Naturally, the signalling level can not be reduced more than $M = 2$. We consider modifying channel coding and in particular, the convolutional encoder. Having performed some tests we found that hardly an improvement can be achieved. In the best case scenarios, a maximum of 0.1dB power reduction can be accomplished at the cost of greater complexity (because of $C > 5$), or a maximum of 0.21dB improvement at the cost of more bandwidth (because of a rate $k/n < 1/2$). The dB reductions concern bit error probabilities (BER) between $10^{-4}$ and $10^{-5}$.

We could definitely not consider the case of not employing interleaving. Swapping its place with channel coding would not work either, since interleaving is mostly effective when used with redundant bit or symbol sequences. What we could consider is the insertion of more redundancy through the use

of a second convolutional encoder after interleaving. In other words, serial concatenation of two channel encoders separated by an interleaver.

Scheme 11 displayed in Fig. 5.2-8 is a typical example. We have chosen the $(3,1,2)$ convolutional encoder shown in Fig. 4.2-1 because of its low constraint length and good error correction properties $(d_{FD} = 5)$. Its performance in comparison to scheme 10 is displayed in Fig. 5.2-9 to 5.2-16. We study the $BER$ performance for two cases. The first case involves the protected bits of Class 1 and the second case involves the total bits, including the unprotected of Class 2. Fig. 5.2-9 to 5.2-12 illustrate the performance for each of the four equivalent discrete models regarding the first case. Fig. 5.2-13 to 5.2-16 illustrate the performance for the four equivalent discrete models regarding the second case.

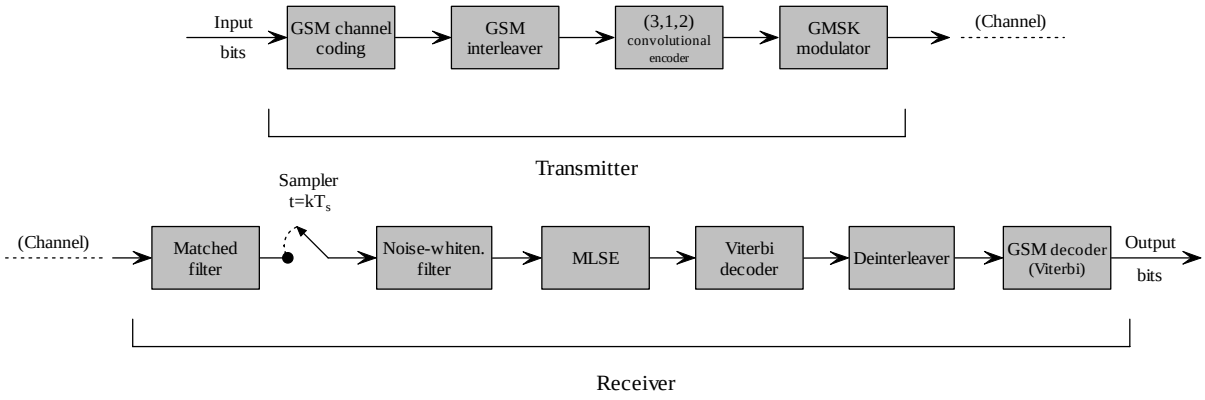


**FIGURE 5.2-8**
Inserting more redundancy in GSM standard (scheme 11).

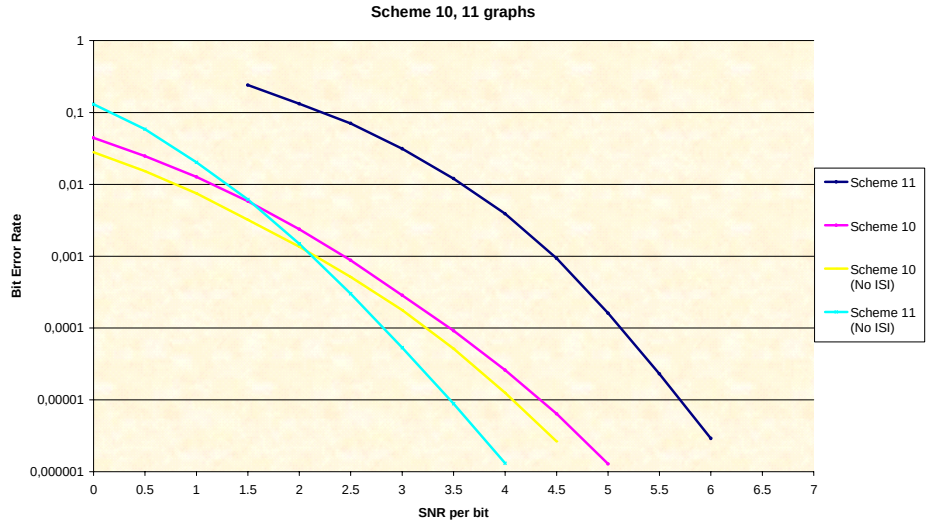


**FIGURE 5.2-9**
Scheme 11 performance and scheme 10 performance for eq. discrete model 1 (protected bits).

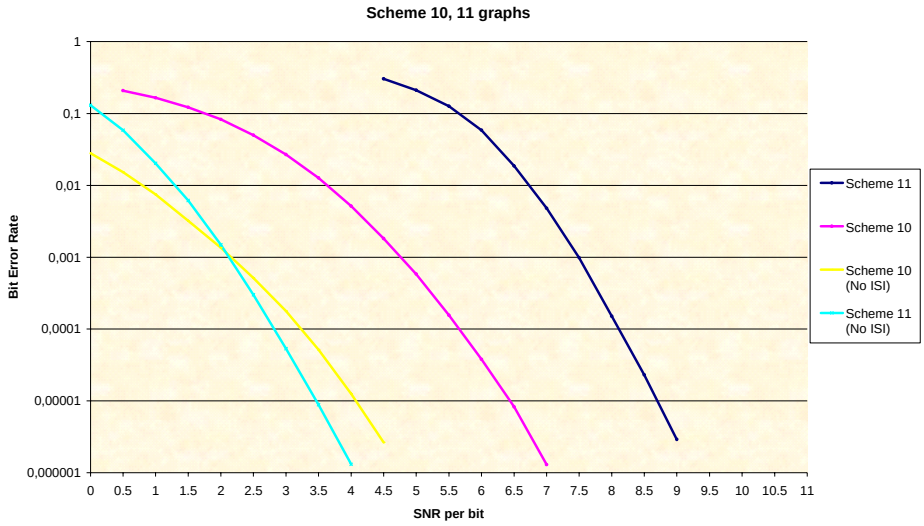


**FIGURE 5.2-10**
Scheme 11 performance and scheme 10 performance for eq. discrete model 2 (protected bits).

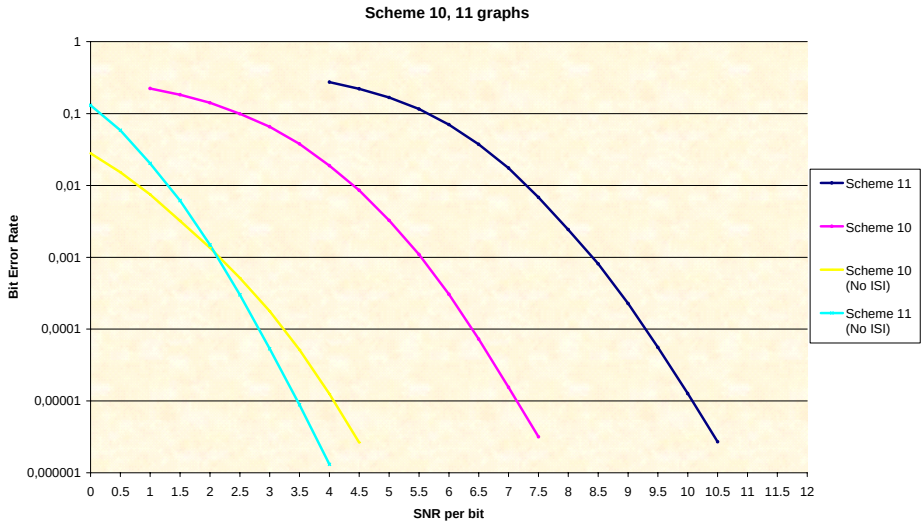


**FIGURE 5.2-11**
Scheme 11 performance and scheme 10 performance for eq. discrete model 3 (protected bits).

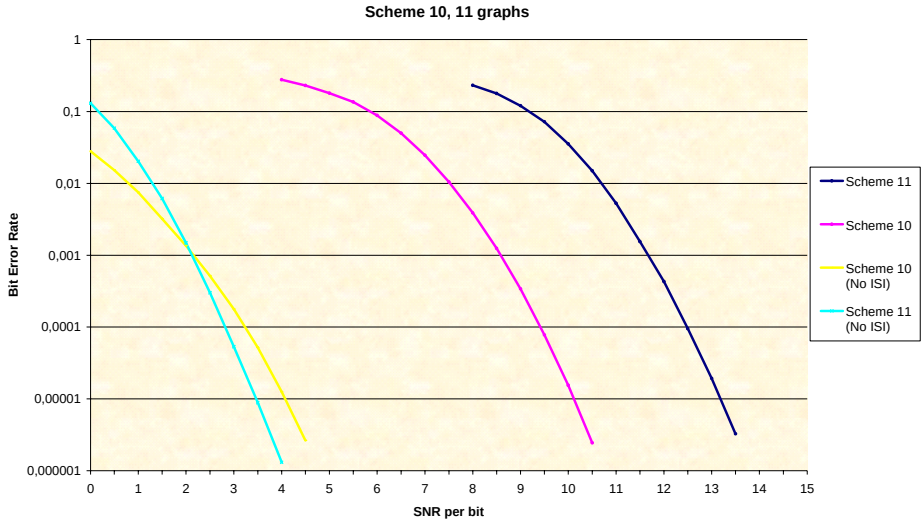


**FIGURE 5.2-12**
Scheme 11 performance and scheme 10 performance for eq. discrete model 4 (protected bits).

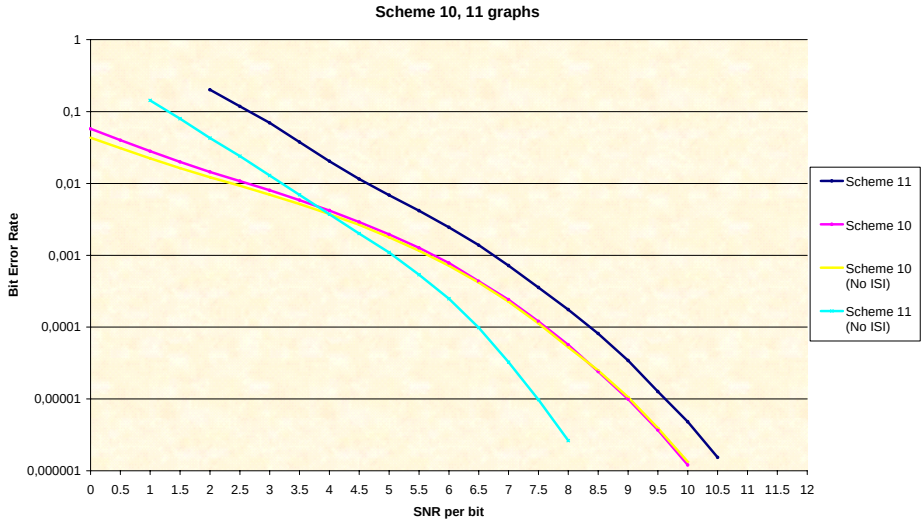


**FIGURE 5.2-13**
Scheme 11 performance and scheme 10 performance for eq. discrete model 1 (total bits).

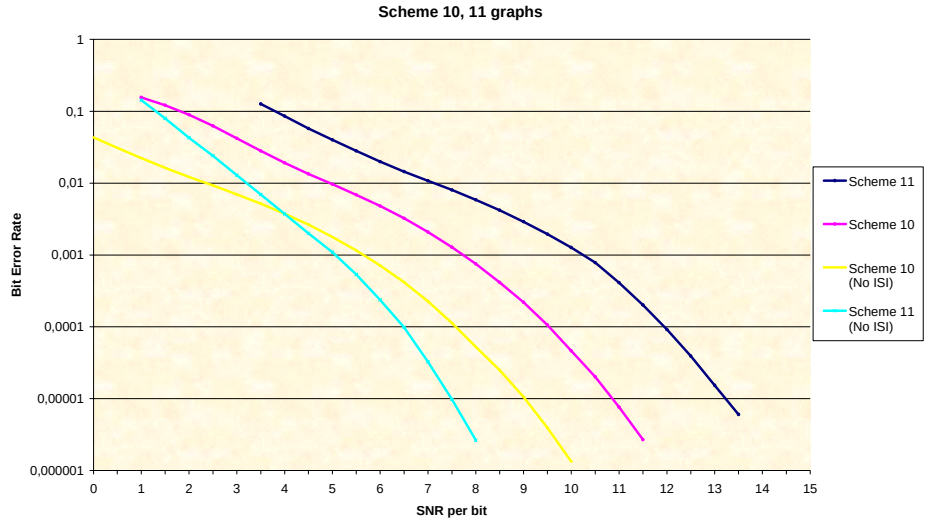


**FIGURE 5.2-14**
Scheme 11 performance and scheme 10 performance for eq. discrete model 2 (total bits).

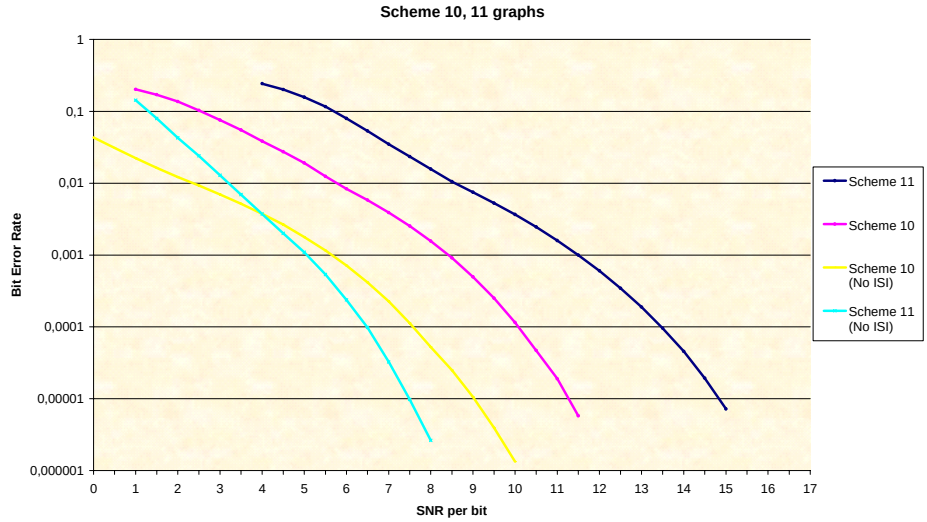


**FIGURE 5.2-15**
Scheme 11 performance and scheme 10 performance for eq. discrete model 3 (total bits).

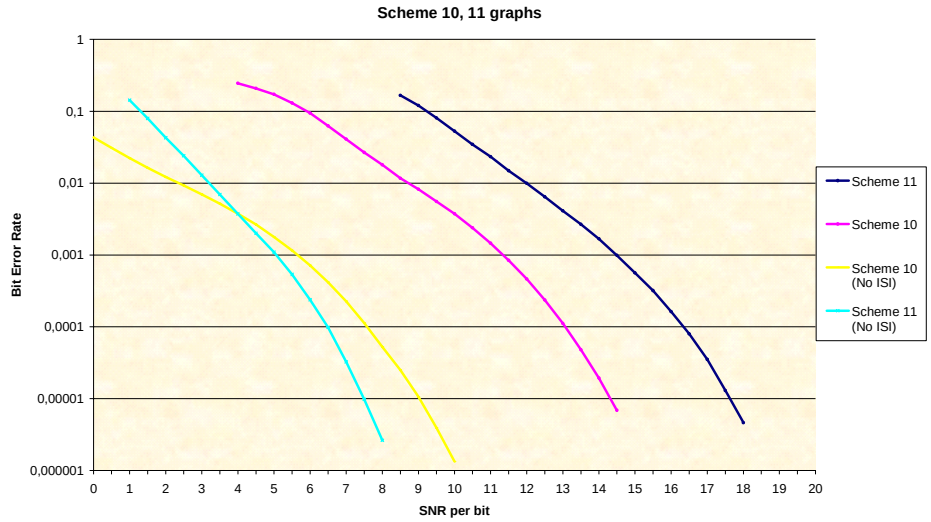


**FIGURE 5.2-16**
Scheme 11 performance and scheme 10 performance for eq. discrete model 4 (total bits).

In order to maintain the same information bit rate, scheme 11 wastes twice the bandwidth of scheme 10. This means that the channel characteristics introduce more ISI. It is taken into account in the above graph results. Consequently, the equivalent discrete model mentioned in each graph refers to the standard case of scheme 10. As can be seen from the graphs, not much of an improvement is accomplished. Concerning the protected bits, scheme 11 requires $1.35$ dB more power than scheme 10 in order to achieve only one error for every $100{,}000$ bits in the case of model 1. Similarly, $2.3$ dB more needed for model 2 and approximately $3$ dB more for models 3 and 4. The only case where scheme 11 surpasses scheme 10 is when there is very much additive noise and no ISI, a rare case indeed. In the absence of ISI, scheme 11 is better when $E_b / N_0$ is less than $2$ dB. That could either imply excessive noise or overwhelming power decrease in the transmission. Concerning an error rate of $10^{-5}$ in the total bits, scheme 11 is $0.6$ dB worse for model 1, $2.4$ dB worse for model 2, $3.5$ dB worse for model 3 and $3.3$ dB worse for model 4. We note for one more time that model 3 is a little more “disastrous” than model 4. We also note that the curves in Fig. 5.2-13 to 5.2-16 drop more linearly in the beginning before dropping fast thereafter. This has to do with class 2 bits which do not pass through GSM channel coding. Many class 2 bit errors survive during the process, especially for low and medium SNR values. This has an effect on the number of total bit errors which reduces exponentially only after a point.

## 5.3 Utilising MS decoding

Fig. 5.3-1 displays scheme 12. It is similar to scheme 11 with the difference that MS decoder replaces MLSE and the following Viterbi decoder in the receiver. As always, GMSK modulator operates with two binary PAM signals, transmitting two bits per symbol. Twice the bandwidth is needed, in comparison with scheme 10, so as to keep up with the information bit rate. We perform simulations for both class1 and class 2 bits for all four channel models in order to have a complete picture. Results can be seen in Fig. 5.3-2 to 5.3-9.

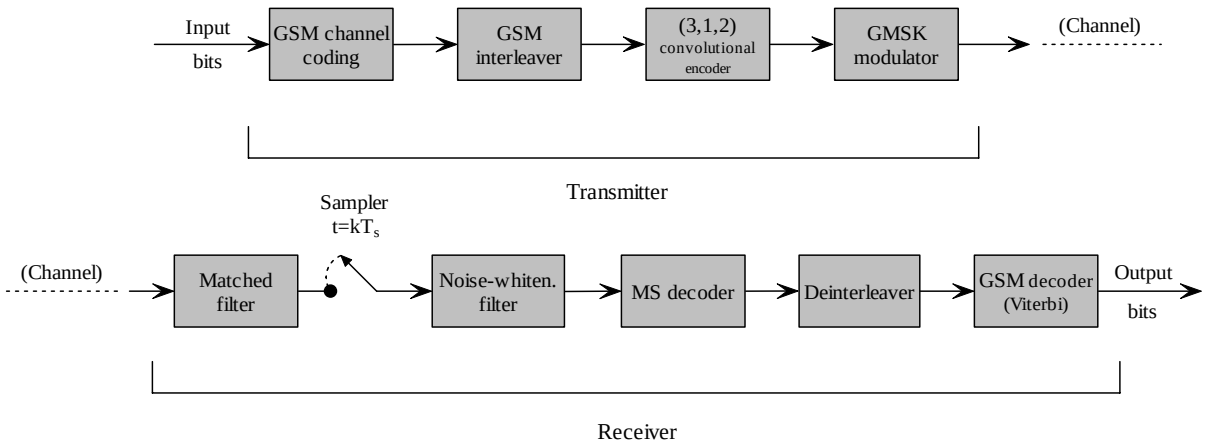


**FIGURE 5.3-1**
Inserting more redundancy in GSM standard and using MS decoder (scheme 12).

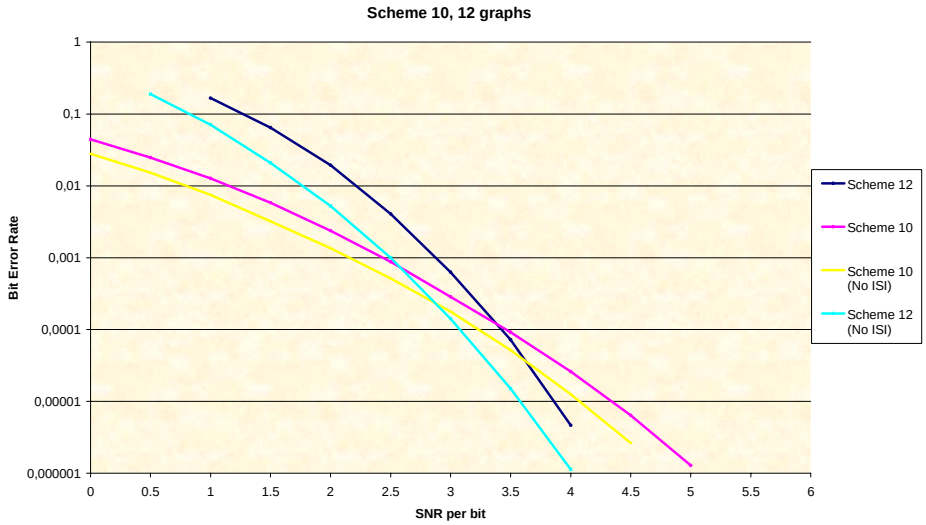


**FIGURE 5.3-2**
Scheme 12 performance and scheme 10 performance for eq. discrete model 1 (protected bits).

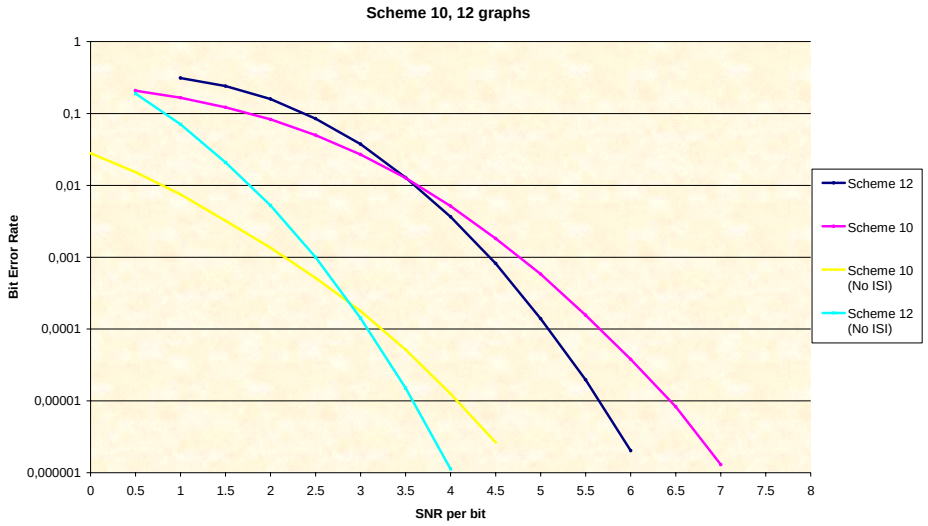


**FIGURE 5.3-3**
Scheme 12 performance and scheme 10 performance for eq. discrete model 2 (protected bits).

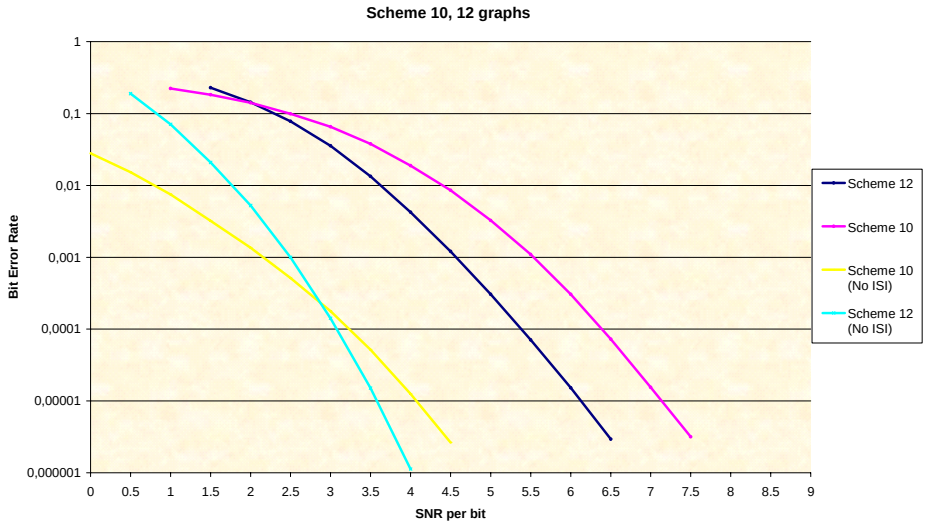


**FIGURE 5.3-4**
Scheme 12 performance and scheme 10 performance for eq. discrete model 3 (protected bits).

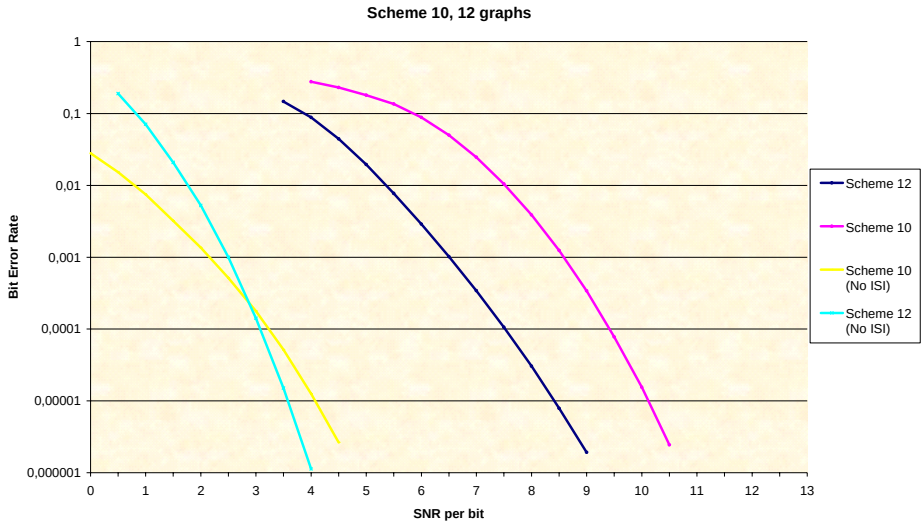


**FIGURE 5.3-5**
Scheme 12 performance and scheme 10 performance for eq. discrete model 4 (protected bits).

In contrast to the disappointing results of scheme 11 in the previous section, scheme 12 performs much better. For a $BER = 10^{-5}$ scheme 12 improves power requirements by $0.5$ dB for model 1, by $0.8$ dB for model 2, by $1$ dB for model 3 and $1.8$ dB for model 4. The more ISI interferes, the more effective MS decoder is. Performance for model 2 is better than performance for model 1, while performance for model 3 is better than performance for model 2 and so on. In general, in low or medium ISI scenarios, scheme 12 produces better results than GSM when SNR is high enough to support the bit error rate under a certain threshold. For models 1 and 2 the thresholds are $10^{-4}$ and $10^{-2}$ respectively. In high ISI scenarios such as models 3 and 4, scheme 12 functions beneficially for almost every SNR.

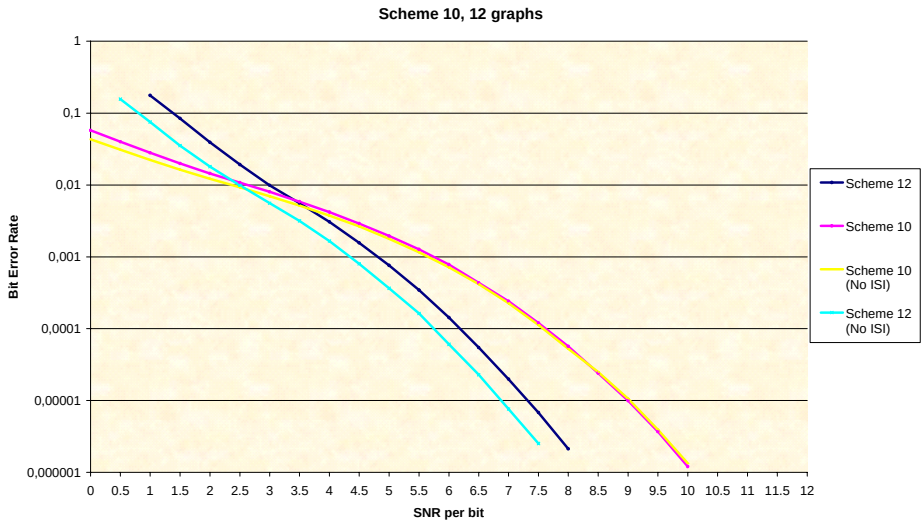


**FIGURE 5.3-6**
Scheme 12 performance and scheme 10 performance for eq. discrete model 1 (total bits).

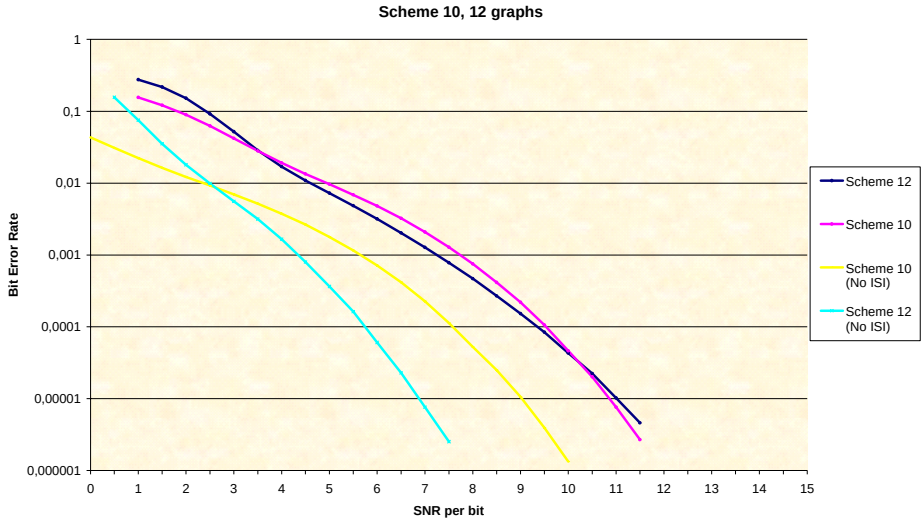


**FIGURE 5.3-7**
Scheme 12 performance and scheme 10 performance for eq. discrete model 2 (total bits).

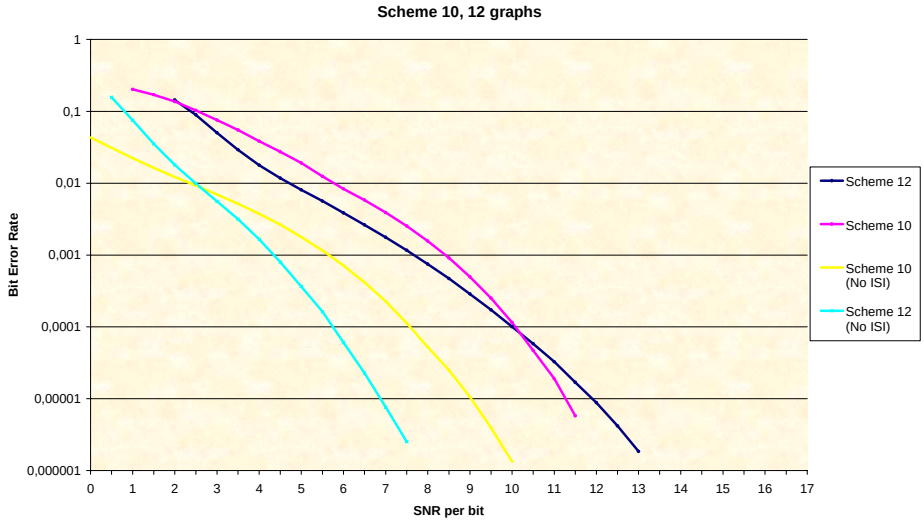


**FIGURE 5.3-8**
Scheme 12 performance and scheme 10 performance for eq. discrete model 3 (total bits).

As can be seen from Fig. 5.3-6 to Fig. 5.3-9, scheme 12 performance is not absolutely superior to that of scheme 10, as far as the total bits are concerned. With the exception of model 1, scheme 12 produces more total errors for the other three models near a $10^{-5}$ bit error rate. This is due to the class 2 bits which do not pass through GSM channel coding. In our analytical simulation sheets, we noticed that the big difference in the class 1 errors (displayed in Fig. 5.3-2 to Fig. 5.3-5) is caused by the combination of MS and GSM channel decoding. Class 2 bits are only "partially" corrected by MS decoder and this alone does not improve class 2 errors compared to scheme 10. Nevertheless, we notice that for total-bit error rates between $10^{-2}$ and $10^{-4}$ scheme 12 exhibits most of the times better performance; however, these error rates are not of primary interest. Only when no or little ISI is involved, is scheme 12 clearly better than scheme 10 for a $BER = 10^{-5}$ (total bits).

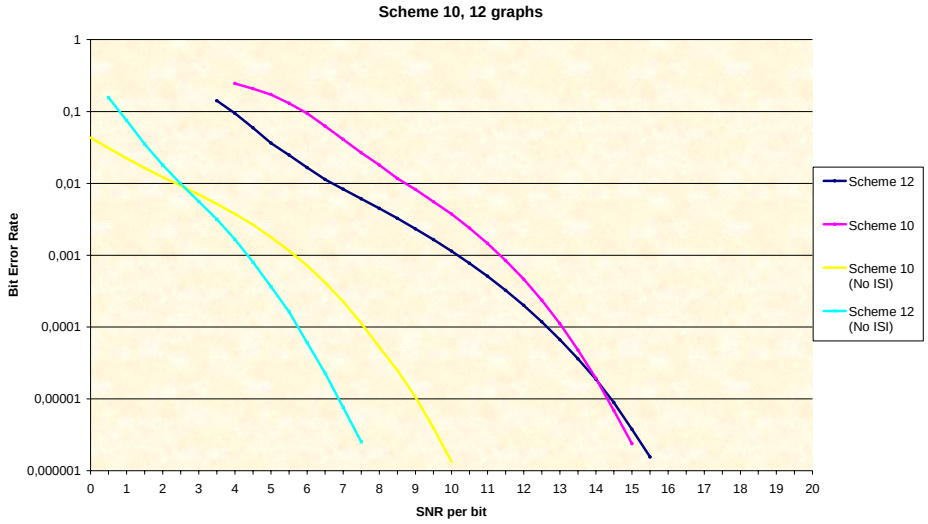


**FIGURE 5.3-9**
Scheme 12 performance and scheme 10 performance for eq. discrete model 4 (total bits).

For a low *BER*, scheme 12 generates fewer class 1 bits in error than scheme 10 at the cost of extra bandwidth. In scheme 13 displayed below we use higher modulation in order to avoid the bandwidth issue. We also define another scheme, namely scheme 14. For convenience reasons, we do not present the block diagram. It is the same with scheme 13 with the difference that it uses the $(5,1,2)$ encoder of Fig. 5.2-4 instead of the $(3,1,2)$ encoder inside the transmitter. MS decoder of scheme 14 has four times the complexity of MS decoder of scheme 13. We are interested to see if the results worth the additional computational burden. We emphasise on the protected bits which are most sensitive to errors.

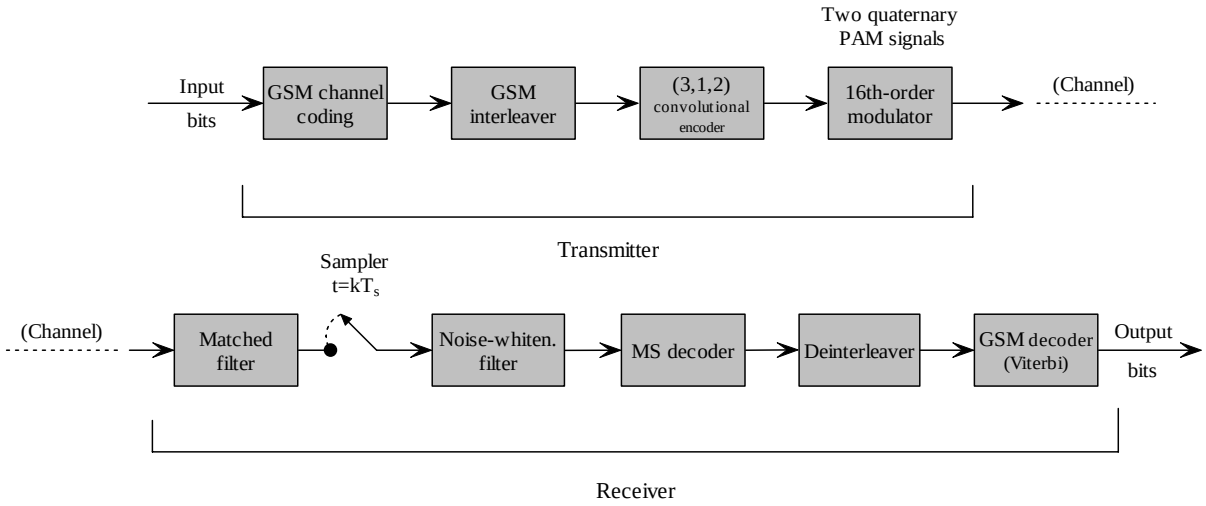


**FIGURE 5.3-10**
Inserting more redundancy in GSM standard and using MS decoder (scheme 13).

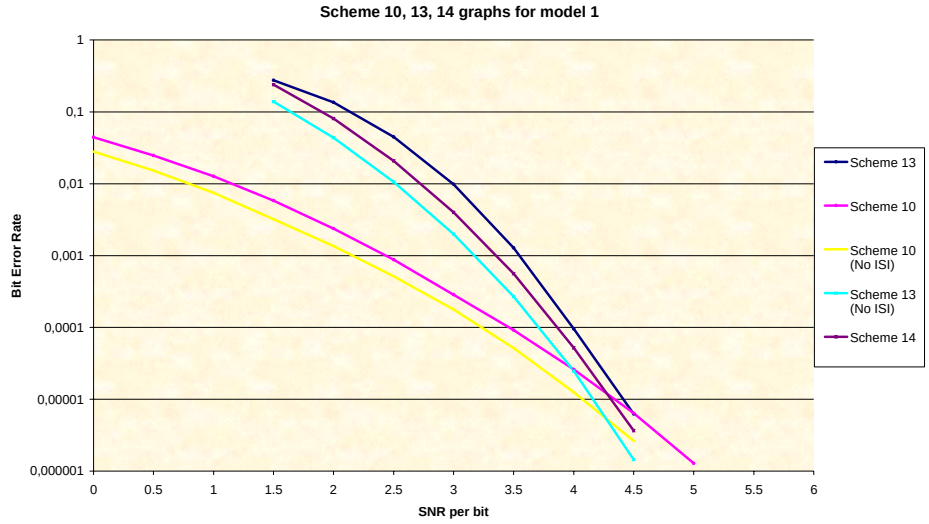


**FIGURE 5.3-11**
Scheme 10, 13 and 14 performance for eq. discrete model 1 (protected bits).

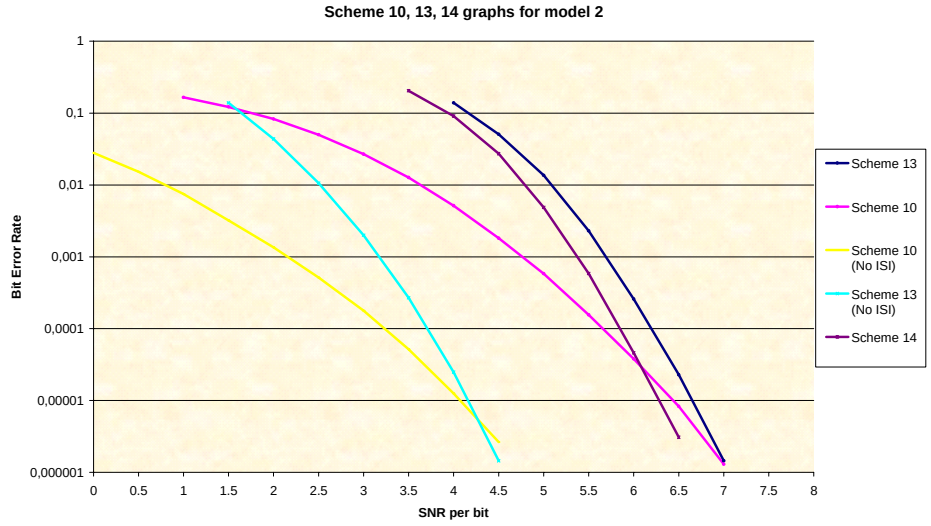


**FIGURE 5.3-12**
Scheme 10, 13 and 14 performance for eq. discrete model 2 (protected bits).

Concerning the first two models, we observe that the higher modulation of schemes 13 and 14 has its effect. Performance is generally worse with only some exceptions. For scheme 13, the exception is when the desired bit error rate is lower than $8\times10^{-6}$ for model 1 and lower than $1.5\times10^{-6}$ for model 2. For scheme 14, the exception has to do with error rates less than $2\times10^{-5}$ for model 1 and less than $5\times10^{-5}$ for model 2. As far as bigger error rates are concerned, both schemes perform worse than the standard case of scheme 10. We note however, that performance of scheme 14 is clearly better than performance of scheme 13.

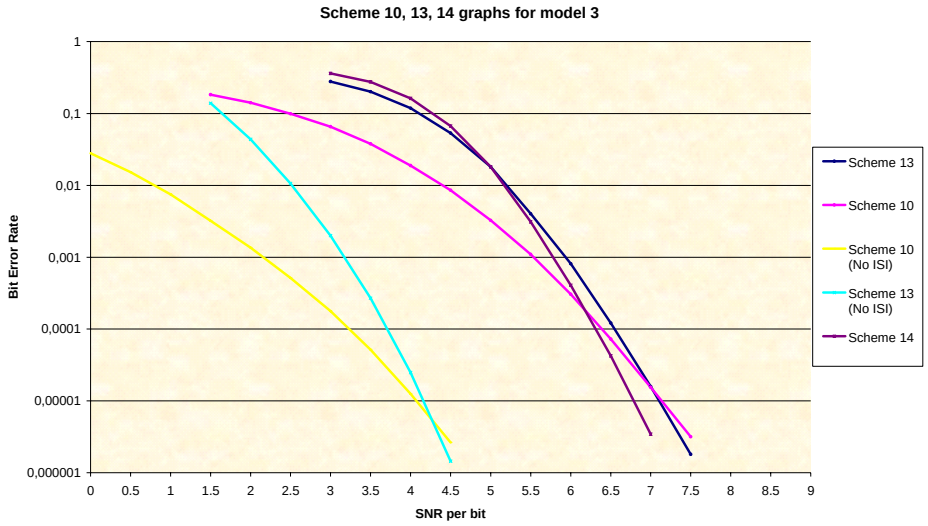


**FIGURE 5.3-13**
Scheme 10, 13 and 14 performance for eq. discrete model 3 (protected bits).

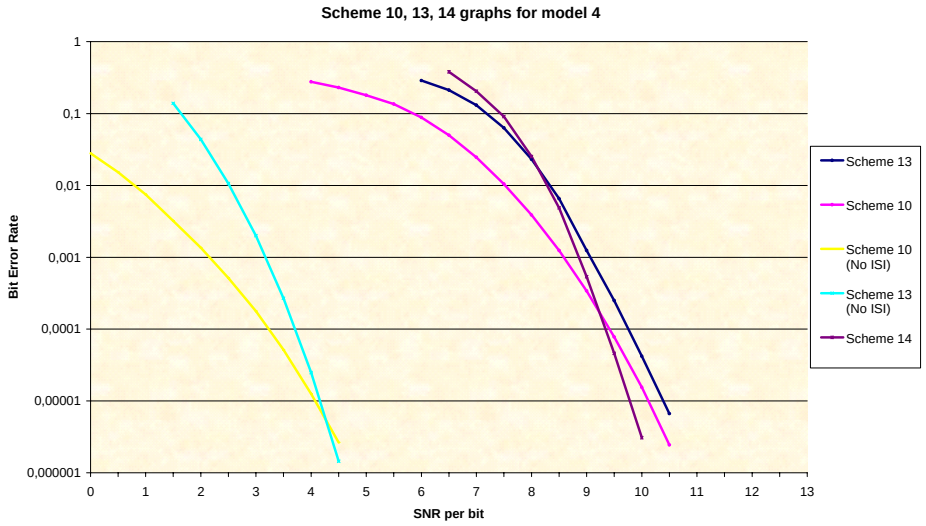


**FIGURE 5.3-14**
Scheme 10, 13 and 14 performance for eq. discrete model 4 (protected bits).

In this case, MS decoder proves to be more beneficial when excessive ISI is involved, such as the case of models 3 and 4. Schemes 13 and 14 exhibit better performance here than in Fig. 5.3-11 and Fig. 5.3-12. As we see, scheme 13 fails to compete against scheme 10 for model 4. It is better only for error rates lower than $10^{-5}$ for model 3. On the contrary, scheme 14 has supremacy over the standard case. It consumes less power for error rates less than $10^{-4}$ for both models 3 and 4. For these models, scheme 14 is worse than scheme 13 regarding greater than $5\times10^{-2}$ error rates.

## 5.4 Additional issues

### 5.4.1 AWGN in digital simulation

Performance graphs displayed in this thesis have been generated by entirely discrete simulation with use of an equivalent discrete model. No adaptive techniques were used for channel estimation in any comparable case. Execution was carried out with the help of a Visual C++ 6.0 compiler. One of the issues encountered was construction of AWGN. For a computer, it is not easy to generate completely random numbers. In addition, the distribution of these random numbers needs to be taken into consideration, in spite of the fact that we are mostly interested in the differences between performances of two or more comparable schemes than the actual performance of a scheme itself. In order to generate normally distributed float numbers, *Box-Muller* method was employed which produces two normally distributed deviates from two uniformly distributed deviates:

```
// Construction of a normally distributed deviate with zero mean and unit variance, via
// the use of Box-Muller method. Function rand( ) can be used as the source of the
// uniformly distributed variables

double CChannel::Gaussian( void)
{
	double unirandom1, unirandom2;
	static int iset = 0;
	static double gset;
	double fac, rsq, v1, v2;

	if (iset == 0)
	{
		do {
			unirandom1 = 1.0*rand()/RAND_MAX;
			unirandom2 = 1.0*rand()/RAND_MAX;
			v1 = 2.0*unirandom1-1.0;
			v2 = 2.0*unirandom2-1.0;
			rsq = v1*v1 + v2*v2;
		} while (rsq >= 1.0 || rsq == 0.0);
		fac = sqrt(-2.0*log(rsq)/rsq);
		gset = v1*fac;
		iset = 1;
		return v2*fac;
	}
	else
	{
		iset = 0;
		return gset;
	}
}
```

C rand( ) function can be used as the source of the uniform distribution. For noise samples having a size of the order of $10^3$ or $10^4$ values, no problem exists. However, for bigger number calls there is a potential case of misinterpreting reality circumstances, because of the limited periodicity of rand( ) function. In addition, random deviates generated by rand( ) function can only be assigned 32,767 different values and this is not 100% realistic for a simulation requiring more than $10^5$ noise samples. Hence, in order to be absolutely certain, *Park & Miller* generator for uniform distributions was used, in place of the rand( ) function. This routine is known to pass all statistical tests except when the number of calls is bigger than the period. The period is of the order of $10^8$. More about Box-Muller and Park & Miller generators as well as random distribution generation of even bigger periods, can be found in [21].

### 5.4.2 Different 4-PAM mapping

In all schemes that have employed quaternary PAM so far, typical {±A, ±3A} mapping was used. From equation (4.5) we see that for this mapping, average power is equal to five. It is five times as much as the average power of $2$ - level PAM. In other words, an increase in the PAM signalling level causes exponential power growth. That is logical since power is always related to the square of the amplitude of a quantity. However, it is quite catastrophic and "expensive" in terms of the $E_b / N_0$ the designer has to pay.

We try to see if it is possible to have a performance improvement through a differentiated mapping. A choice such as that of Fig. 5.4-1, would definitely reduce the average power of the $4$-PAM signal. Indeed, average power $S$ for the mapping of Fig. 5.4-1 is $S = 2.5,$ half of the power of the typical $4$-PAM mapping. The cost is that the distance properties of the signal decrease, and detection is more susceptible to errors.

For example, let us consider encoder of Fig. 4.2-1 which is used within scheme 13. Looking at its Trellis in Fig. 4.2-3, we see that for the typical mapping, $d_{FED} = 36.$ For the differentiated mapping of Fig. 5.4-1, $d_{FED} = 19.$ With the help of simulation results in Fig. 5.4-2 and Fig. 5.4-3 we investigate whether the power decrease benefits overcome the reduction of $d_{FED}$. We use scheme 13 as a reference. Scheme 13b is identical to scheme 13, with the only difference that the below $4$ - PAM mapping is used.

$"11" \rightarrow 2A$
$"10" \rightarrow A$
$"00" \rightarrow -A$
$"01" \rightarrow -2A$

**FIGURE 5.4-1**
4-PAM mapping for scheme 13b.

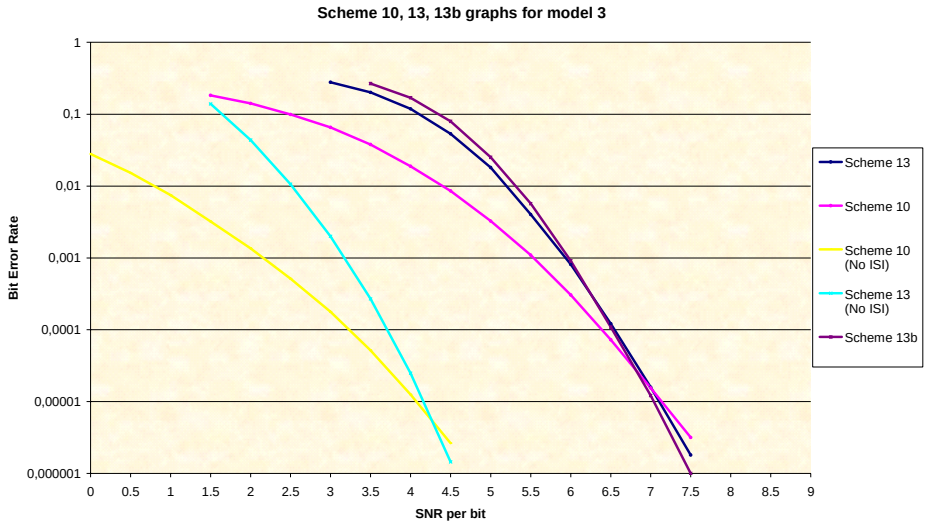


**FIGURE 5.4-2**
Scheme 10, 13 and 13b performance for eq. discrete model 3 (protected bits).

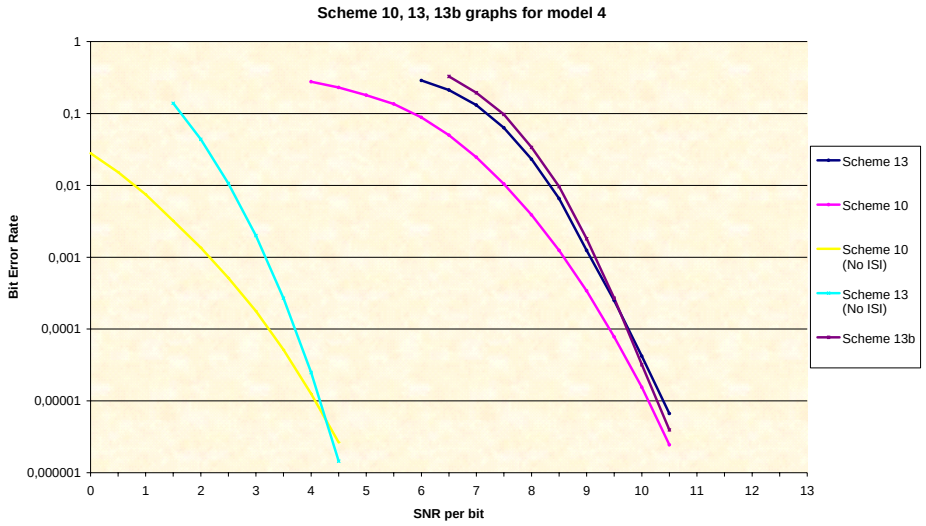


**FIGURE 5.4-3**
Scheme 10, 13 and 13b performance for eq. discrete model 4 (protected bits).

As can be seen from the graphs, performance does improve for bit error rates lower than $10^{-4}$. It worsens for relatively high error rates. This is reasonable and expected. For low SNRs where noise values are relatively high (compared to signal values), many errors are delivered to MS decoder. Reduced $d_{FED}=19$ cannot "handle" these excessive errors as well as $d_{FED}=36$ of scheme 13. On the other hand, for high SNRs, only a few errors enter MS decoder. These few errors can be detected and satisfactorily corrected both by $d_{FED}=19$ of scheme 13b and by $d_{FED}=36$ of scheme 13. Therefore we conclude that a different mapping can have a positive effect on performance, depending on the desired BER.

We could ask ourselves in what deal we are able to improve power requirements for scheme 13, by further differentiating the $4$ - PAM mapping. We noticed in Fig. 5.4-3 that changing the PAM symbol mapping from {±A, ±3A} to {±A, ±2A} resulted in $0.1$ dB improvement for a $BER=10^{-5}$. We are looking for the "best" $4$ - PAM mapping in order to achieve even greater $E_b/N_0$ reduction by further modifications.

We consider a general form of $4$ - PAM mapping as in Fig. 5.4-4 below with $x\in\Re$ and $A, y\in\Re^*$.[1]

$"11"\rightarrow(x+y)A$
$"10"\rightarrow yA$
$"00"\rightarrow -yA$
$"01"\rightarrow -(x+y)A$

**FIGURE 5.4-4**
General 4-PAM mapping.

We are interested in determining the relation between $x,\ y$ so that the ratio

[1] If $y$ was null then we would not have to do with $4$ - PAM but with $2$ - PAM.

$$\frac{d_{FED}}{S} = \frac{2(Ax + 2Ay)^2 + A^2x^2}{\frac{(Ax + Ay)^2 + A^2y^2}{2}} \tag{5.10}$$

is maximum. We know that $A, y \neq 0.$ If we set $u = \frac{x}{y} \in \Re,$ it is enough to find the maximum of the function

$$f(u) = \frac{4(u+2)^2 + 2u^2}{(u+1)^2 + 1} \tag{5.11}$$

We notice that $\frac{df}{du} = 0 \Leftrightarrow u \in \{0, -2\}.$ The first value indicates that $x = 0$ and the second value that $x = -2y.$ Both solutions correspond to the same binary PAM mapping {+A, -A}. This means that the closer symbol "11" to symbol "10" and symbol "01" to symbol "00" are, the greater the ratio $d_{FED}/S$ becomes. This is reasonable since the closer the external symbols "11" and "01" are placed near the internal symbols "10" and "00" the more we have to with binary signalling instead of quaternary. However, it has been observed by simulations that any mapping with $x$ much smaller than $yA$ yields indeed better results, but only for extremely low bit error rates (under $10^{-6}$ or $10^{-7}$). Not many communication standards emphasise on achieving so low a probability of error.

As a result, for a bit error rate near $10^{-5},$ the best 4-PAM mapping is the one that assigns to the more distant symbols, values close to $2yA$ and $-2yA$. Such a mapping is displayed in Fig. 5.4-1. It is a sub-case of mapping of Fig. 5.4-4 for $x = y = 1.$ We summarise that with $4$-PAM mapping of Fig. 5.4-1, performance of schemes 4, 5, 13, 14 can be slightly improved as far as a bit error rate near $10^{-5}$ is concerned.

# 6. CONCLUSIONS

## 6.1 General summary

MS decoder presented in this thesis is the overall optimum receiving method of a Trellis-coded modulated signal in a multipath environment. It combines both equalising and convolutional decoding in a single unified process based on the relative Trellis diagram. In a few words, it exploits dependencies introduced by the encoding process in order to conduct more efficient equalisation within the unified process. More efficient equalisation produces less erroneous results and makes the task of decoding also easier, since the minimum possible information is "lost" through out the detection operation.

Concerning the computational and design complexity of employing MS decoder, no significant additional burden was noticed. The important thing is that any extra encumbrance does not depend on the channel characteristics. It only depends on the properties of the convolutional encoder that is MS-decoded in reception (reference encoder). We distinguish two main categories. Employing MS decoder without any bandwidth increase and employing it with bandwidth increase. In the first case, complexity never exceeds $2^{k(C-1)}$ times the complexity of the ML detector, where $k, C$ are the input bits and constraint length of the inserted reference encoder. In many cases, complexity decreases rather than increases as well. In the second case, complexity decreases or increases depending on the case. However, any increase rarely exceeds the predefined limit of $2^{k(C+1)}$ times the complexity of the ML detector, where $k, C$ are the input bits and constraint length of the inserted reference encoder.

There are many ways to employ MS decoder in a standard. It always depends on the goals of the communication system and on where the manufacturer wants to emphasise. In a balanced standard, where bandwidth, information bit rate, complexity and power are all important, relations between these factors are usually tight. Any attempt to upgrade one of the factors implies worsening another factor, which is undesirable. For instance, any change in order to improve power requirements leads to an increase on bandwidth and the opposite. MS decoder succeeds in loosening these tight dependencies giving more space for improvement. Schemes 3 and 4 show a good case in point. As can be seen from Fig. 3.4-13 to Fig. 3.4-16, with no extra bandwidth and no less information bit rate, power improvement is achieved. The price that has to be paid is $16$ times more computations and storing of paths and metrics than the standard scenario and that referring only to one of the executed operations. It is a rather insignificant load, considering the large power decrease that is achieved and considering the rate with which today processor capabilities advance.

Another good case in point is displayed on Fig. 5.3-2 to Fig. 5.3-5. We see that a considerable power reduction is accomplished at the cost of double bandwidth. This can be useful for the creation of a healthier mobile standard. Without the use of MS decoder results are worse both for power and bandwidth as can be seen from Fig. 5.2-9 to Fig. 5.2-12. We emphasise at the protected bits of GSM standard, since those are of primary interest. Results in total bits are not enhanced (Fig. 5.3-6 to Fig. 5.3-9), but a better transmission quality is guaranteed because of the fewer GSM class1 bits in error.

One more typical example is scheme 14. The relative results are displayed in Fig. 5.3-11 to Fig. 5.3-14. As can be seen, its performance compared with GSM (scheme 10) is superior for low bit error rates. For higher bite error rates, scheme 14 demands more power. Considering the fact that the standards are intended for mobile communication, scheme 14 might be preferable from a manufacturer. The reason is that the maximum power is reduced. Although mathematically mean power (for all bit error rates) remains the same or even increases, what is important is power on the range $10^{-4} < BER < 10^{-6}$. Most communications that do not suffer from a signal loss, take place in that area. In other words, the SNR range covered by curve of scheme 14 is smaller and with a lower maximum value that implies decreased maximum radiation.

It was also found that schemes employing MS decoding like scheme 14 could be further improved by differentiation in the PAM amplitude levels. Scheme 14 uses $16$- th order modulation and as a result, $4$- level PAM signalling. By replacing the traditional $\{\pm A, \pm 3A\}$ $4$- PAM by a $\{\pm A, \pm 2A\}$ amplitude assignment, we achieve a further slight enhancement for the crucial low bit error rates. As far as symbol mapping is concerned, mapping with the method of set partitioning devised by Ungerboeck is recommended. In this way, the distance between converging Trellis paths increases and MS decoder is less likely to "confuse" which of the paths is correct. Susceptibility to errors is consequently shortened.

The issue of interleaving was also considered. In wireless applications, interleaving is indispensable in order to deal with the consecutive burst errors caused by multipath fading or other factors. The conclusion is that interleaving cannot be used after the reference encoder, if this encoder is to be decoded by an MS decoder in the receiver. Interleaving eliminates any dependencies introduced by the reference encoder. Without these dependencies, MS decoder provides no better results than an ML detector. Naturally, interleaving and other operations can be employed before the reference encoder in the transmitter.

On the whole, it is mentioned that MS decoding can be used not only in mobile communication, but in other kinds of wireless applications as well. In particular, it can be used in any case where a convolutional encoder is employed. If the reference encoder is placed before modulator then the results are significantly improved; if not, then MS decoder functions in the same way as an MLSE and the results are typically the same. A good way of exploiting MS decoder is to use it in order to reduce power demands. Satellite communication is a good example. With no strict bandwidth limitations, a satellite standard could be improved so as to reduce necessary power and increase transponder life duration. Nevertheless, because of the generality of MS decoder, different uses can be devised and various standards can be potentially altered. In this thesis, we focused on GSM 3GPP standard.

## 6.2 Further work

The issues mentioned in this section are ideas of potential MS decoder use or improvement and in no way limit the range of applications that can be considered. Throughout this thesis we have dealt with convolutionally-coded sequences or sequences encoded by similar means such as encoder of Fig. 3.4-18. Considering the fact that encoders similar to Trellis encoders are used in many applications, it would be practically interesting to employ MS decoder for such non-linear convolutional encoders. An example is shown in Fig. 6.2-1. It is a non-linear eight-state convolutional encoder. It is used before a $32$- QAM modulator operating with a rectangular signal constellation that is invariant under $90^\circ$ phase rotations [38]. This code has been adopted as a standard (V.32 and V.33) for $9.6$ Kpbs and $14$ Kpbs telephone line modems.

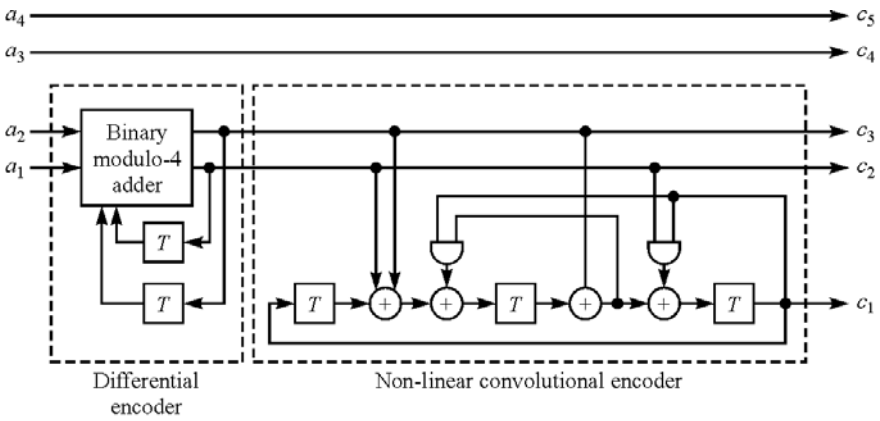


**FIGURE 6.2-1**
Eight-state non-linear convolutional encoder for a $32$-QAM signal constellation.

Although marketing issues are not a topic of this work, it has to be mentioned that standardisation with the help of MS decoder could lead to a healthier mobile telephony standard. Even if there is a considerable price to pay (e.g. double bandwidth), such an approach would still be commercially interesting. From the manufacturer's point of view, much of the capital saved from less power usage would be spent on increased bandwidth. The users however, would be willing to pay much more for mobile communication that is proved to be less harmful. In addition, the cost to obtain bandwidth resources in any state is paid dearly but only once. The energy cost for the antennas operation is paid on the other hand, constantly. Nevertheless, in the writer's opinion all of these potential improvements to GSM 3GPP standard have to do with mobile communication as far as the next decade is concerned. After that, the existing standards are going to be replaced by new technologies employing OFDM and operating at more than 100 Mbps information bit rate. One of the biggest operators in Japan has already designed relative standards, successfully produced mobile devices and simulated such performances.

Another worth mentioning future case study would be alteration of MS decoder so as to produce "soft" output and to work with feedback information from a following decoder in a standard. This soft information would be Log-Likelihood-Ratios (LLRs) as defined in (2.14). Fig. 6.2-2 displays an example. MS decoder receives the input symbol sequence from the Noise-whitening filter. It computes the logarithm of the likelihood ratio of the coded bits and sends it to the deinterleaver. MAP decoder receives the deinterleaved logarithms and based on them, computes two things: The LLRs of the information bits (that served as an input to the encoder that is decoded with MAP decoder) and the LLRs of the coded bits. The LLRs of the coded bits that served as input to MAP decoder are substracted from the LLRs of the coded bits produced by MAP decoder and the result is interleaved. It is called *extrinsic* information and is fed back to the MS decoder. The cycle repeats one or more times with the difference that now only the extrinsic part of the MS decoder output is deinterleaved and fed to MAP decoder. With simple words, we have to do with a structure similar to *Turbo equalisation* but with MS decoder substituting MAP equaliser.

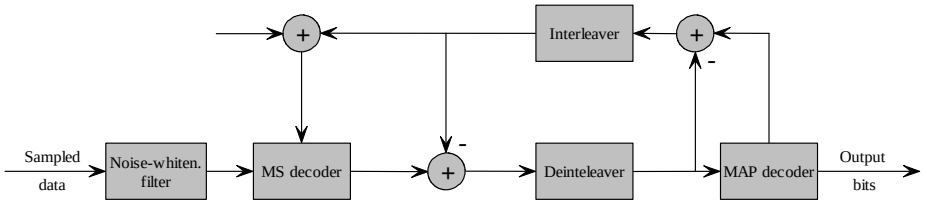


**FIGURE 6.2-2**
Iterative MS decoding and MAP decoding.

With the implementation of such a scheme the benefits are obvious. We notice that redundancy exists in two stages, MS decoding and MAP decoding. Information is considerably "protected" at the cost of extra bandwidth. However, significant error reduction (and consequently power decrease) could be achieved. The reason is that MS decoding will exploit one piece of redundancy in the first pass and both pieces of redundancy in the following iterations. During the iterations, MS decoding will be carried out based on the information bits that served as input to the encoder that is MAP-decoded and not on the bits that served as input to the encoder that is MS-decoded. This utilisation of sampled data based on the very initial input is an important advantage.

In addition to the above, it would be noteworthy to adjust MS decoder operation so as to process *Turbo-coded* signals that are extensively employed in third generation mobiles. Naturally, block coding could also be considered, but its use has to do with only a few standards. Convolutional coding is widely preferred. Last, as far as simulations in a lab environment are concerned, it would be interesting to test MS decoder performance without perfect channel knowledge, i.e. with adaptive channel estimation techniques.

# References - Bibliography


[1] Nyquist H., 1924. *"Certain factors affecting Telegraph Speed"*, Bell Syst. Tech. J., vol. 3, p. 324.

[2] Nyquist H., 1928. *"Certain topics in Telegraph Transmission Theory"*, AIEE Trans., vol. 47, pp. 617-644.

[3] Shannon C. E., 1948. *"A Mathematical Theory of Communication"*, Bell Syst. Tech. J., vol. 27, pp. 379-423, July.

[4] Shannon C. E., 1948. *"A Mathematical Theory of Communication"*, Bell Syst. Tech. J., vol. 27, pp. 623-656, October.

[5] L. N. Thang & R. M. A. P. Rajatheva. *"Performance of Parallel Concatenated Convolutional Codes (Turbo codes) and Serial Concatenated Convolutional Codes in Wideband DS-CDMA"*, retrieved from the World Wide Web in 11/12/2004. www.science.unitn.it/~thang/Publication/etripaper.pdf

[6] L. Litwin, 2001. *"Matched filtering and timing recovery in digital receivers"*, retrieved from the World Wide Web in 07/06/2005. http://rfdesign.com/mag/radio_matched_filtering_timing/

[7] D. Johnson, 2001. *"White Gaussian Noise"*, retrieved from the World Wide Web in 12/02/2005. http://cnx.rice.edu/content/m11281/latest/

[8] H. L. Van Trees, 2001. *"Detection, Estimation, and Modulation Theory, Part I"*, John Wiley & Sons Inc., New York, US.

[9] John M. Wozengraft & B. Reiffen, 1961. *"Sequential Decoding"*, The MIT Press.

[10] Jelena Nikolic-Popovic, 2000. *"Implementing a MAP decoder for cdma2000m Turbo codes on a TMS320C62x DSP device"*, Wireless ASP products, Texas Instruments, retrieved from the World Wide Web in 18/08/2003. http://focus.ti.com/lit/an/spra629/spra629.pdf

[11] Stefan Hoest, 1999. *"On Woven Convolutional Codes (ch. 3)"*, Department of Information Technology, Lund University, Sweden, retrieved from the World Wide Web in 28/10/2004. http://www.it.lth.se/stefanh/Thesis/Thesis.pdf

[12] Bellman, R., 1957. *"Dynamic Programming"*, Princeton University Press, Princeton, New Jersey, US.

[13] Rolf Johannesson, Kamil Sh. Zigangirov, 1999. *"Fundamentals of Convolutional Coding (IEEE Series on Digital & Mobile Communication)"*, The Institute of Electrical & Electronics Engineers Inc., New York, US.

[14] Lajos Hanzo, T. H. Liew, B. L. Yeap, 2002. *"Turbo Coding, Turbo Equalisation and Space-Time Coding for Transmission over Fading Channels"*, John Wiley & Sons Ltd, West Sussex, UK.

[15] John G. Proakis, 2001. *"Digital Communications, 4$^{th}$ edition"*, McGraw-Hill Book Co. – Singapore.

[16] *"Definition on estimators"*, The Johns Hopkins, University Baltimore, US, retrieved from the World Wide Web in 21/07/2004. http://www.econ.jhu.edu/people/shum/stats/lect6.pdf

[17] *"Channel coding; Hamming distance"*, University College London, UK, retrieved from the World Wide Web in 21/07/2004. http://www.cs.ucl.ac.uk/staff/S.Bhatti/D51-notes/node30.html

[18] *"Hamming distance"*, University of Reading, UK, retrieved from the World Wide Web in 21/07/2004. http://www.personal.reading.ac.uk/~sis01xh/teaching/CY2G2/IT6.pdf

[19] G. L. Stuiber, 1994. *"Trellis-codes and adaptive equalization for multipath fading ISI channels"*, School of Electrical and Computer Engineering, Georgia Institute of Technology, US.

[20] R. W. Lucky, 1966. *"Techniques for adaptive equalization of digital communication systems"*, American Telephone and Telegraph Co, US.

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery, 1992. *"Numerical recipes in C"*, Cambridge University Press, Cambridge, UK.

[22] Gérard Terreault, S. T. Broadband, 2002. *"Signal Impairments. A closer look at MER and BER"*, retrieved from the World Wide Web in 14/01/2005. http://www.ct-magazine.com/archives/ct/1202/1202_signal.html

[23] Roger L. Freeman, 1998. *"Telecommunications Transmission Handbook"*, Wiley Series in Telecommunications and Signal Processing, John Wiley & Sons Inc., New York, US.

[24] A. H. Aghvami, 1987. *"16-ary QAM transmission through two-link nonlinear satellite channels employing a baseband linearizer"*, Proceedings IEEE Int. Conf. on Communications, ICC 87, June 1987, pp. 914-918.

[25] A. H. Aghvami, 1984. *"Performance analysis of 16-ary QAM signalling through two-link nonlinear channels in additive Gaussian noise"*, IEE Proceedings, vol. 131, Part F, No. 4, July 1984, pp. 403-406.

[26] J. Penton, W.W. Wu, 1987. *"The challenge of 21st. century satellite comminications: INTELSAT enters the second millenium"*, IEEE Journal on Selected Areas in Communications, vol. SAC-5, No. 4, May 1987, pp. 571-591.

[27] G. Ungerboeck, 1982. *"Channel Coding with Multilevel/Phase Signals"*, IEEE Transactions Information Theory, vol. IT-28, pp. 55-67, January.

[28] G. Ungerboeck, 1987. *"Trellis-Coded Modulation with Redundant Signal Sets, Parts I and II"*, IEEE Commun. Magazine, vol. 25, pp. 5-21, February.

[29] Justin Romberg, 2003. *"Nyquist Theorem"*, retrieved from the World Wide Web in 05/09/2004. http://cnx.rice.edu/content/m10791/latest/

[30] KBS. *"GSM Guide & History"*, retrieved from the World Wide Web in 09/04/2005. http://lebanoncell.com/gsm_guide_and_history.htm

[31] S. Grech, 1999. *"Channel Coding Standards in Mobile Communications"*, retrieved from the World Wide Web in 13/04/2005. http://www.tml.hut.fi/Studies/Tik-110.300/1999/Wireless/channel_1.html

[32] *"GSM 900"*, Electrical & Computer Engineering Department, University of Toronto, Canada, retrieved from the World Wide Web in 25/08/2004. http://www.eecg.toronto.edu/~nazizi/gsm/index.html

[33] S. Haykin, 2001. *"Communication Systems, 4th Edition"*, John Wiley & Sons, Inc., New York, US.

[34] S. Scourias, 1997. *"Overview of the Global System for Mobile Communications"*, retrieved from the World Wide Web in 19/04/2005.
http://ccnga.uwaterloo.ca/~jscouria/GSM/gsmreport.html

[35] J. Siegel-Itzkovich, D. Carrington, 2003. *"GSM phone encryption can be cracked"*, retrieved from the World Wide Web in 04/09/2003.
http://www.newscientist.com/article.ns?id=dn4130

[36] Hugo M. Tullberg, Prof. Paul H. Siegel, *"Interleaving techniques for coded modulation for mobile communications"*, retrieved from the World Wide Web in 27/07/2004. http://cwc.ucsd.edu/~htullber/Text/SOEreviewtext.pdf

[37] Mohammad R. Soleymani, Gao Yingzi, U. Vilaipornsawai, 2002. *"Turbo Coding for Satellite and Wireless Communications"*, Kluwer International Series in Engineering and Computer Science, Kluwer Academic Publishers.

[38] L. F. Wei, 1984. *"Rotationally Invariant Convolutional Channel Coding with Expanded Signal Space"*, IEEE Journal on Selected Areas in Communications, vol. SAC-2, pp. 659-686, September.