SOME NEW RESULTS

ON CONVOLUTIONAL CODES

Volume 2: Appendices

by

Nikos P. Frydas

A Doctoral Thesis

Submitted in partial fulfilment of the requirements

for the award of

Doctor of Philosophy of the Loughborough University of Technology

1 January 1990

Contents

Volume 2

Abbreviations

APP = a posteriori probability (see p. 136)
AWGN = additive white Gaussian noise (see p. 287)
BSC = binary symmetric channel (see p. 6)
CC = convolutional code (see p. 18)
CEG = central group (see p. 58)
CSOC = convolutional self-orthogonal code (see p. 138)
c/w = codeword
DD = definite decoding (see p. 139)
DIG = discarded input group (see p. 87)
DMC = discrete memoryless channel (see p. 5)
EA = encoding array (see p. 220)
eqn = equation
FD = feedback decoding (see p. 139)
FEG = front-end group (see p. 58)
FF = feed-forward (see p. 317)
Fig. = Figure
gcd = greatest common divisor (see p. 435)
IA = initial array (see p. 183)
iff = if, and only if
ING = input group (see p. 87)
I/P = input
LHS = left-hand side
LSB = least significant bit
LSC = linear sequential circuit
MEG = memory group (see p. 58)
MIG = memory input group (see p. 87)
MLD = maximum likelihood decoding (see p. 10)
MSB = most significant bit
MUX = multiplexing (see p. 283)
O/P = output
PCM = pulse code modulation
PSK = phase-shift keying
REG = rear-end group (see p. 58)
reln = relation
RHS = right-hand side
SA = syndrome array (see p. 517)
SNR = signal-to-noise ratio
SO = self orthogonal (see p. 138)
SR = shift register
SYRE = syndrome register (see p. 210)
X-OR = exclusive-or
wrt = with respect to

Notation

$\mathcal{S}$ = = = number of states (Chapter 4) (see p. 89)

$\mathcal{E}$ = = = composite parity-check (Chapter 6) (see p. 135)

β = = = IA generating element (Chapters 7 & 8) (see p. 218)

$C(\tau)$ = = autocovariance function (see p. 257)

$C(n,k) \triangleq n!/[k!(n-k)!]$ = binomial coefficient

Γ = = = signal-to-noise ratio per information-bit

D = = = delay operator

d_{min} = = minimum distance of a code

E = = = energy per received bit (see p. 4)

E[?] = = expected value of ?

e = = = channel-error sequence (see p. 46)

erfc = = complementary error function (see p. 291)

$\mathcal{f}$ = = = coherent demodulator O/P (Chapter 1) (see p. 286)

$\mathcal{f}$ = = = number of zero-length SRs (Chapters 3 & 4) (see p. 57)

$\mathcal{f}(i)$ = = memory-density function (Chapter 4) (see p. 102)

$\mathcal{F}(i)$ = = memory-distribution function (Chapter 4) (see p. 114)

Φ = = = Euler totient (Chapters 7 & 8) (see p. 436)

G = = = net coding-gain (see p. 13)

G = = = generator matrix (see p. 30)

GF(q) = Galois field q (see p. 297)

H = = = parity-check matrix (see p. 45)

I = = = identity matrix

J = = = number of orthogonal check-sums

M = = = total circuit memory (see p. 55)

m = = = memory order (see p. 19)

M_i = = = length of the ith SR of a normal LSC (see p. 33)

$\tilde{n}$ = = = single-sided noise power spectral density

n_A = = = actual constraint-length (see p. 20)

n_E = = = effective constraint-length (see p. 145)

$\underline{o}$ = = = number of input blocks (Chapter 4) (see p. 89)

$P_d, \mathbf{P}_d$ = probability of bit decoding error

P_e = = = probability of channel error

r = = = received sequence (see p. 46)

$R(\tau)$ = = autocorrelation function (see p. 257)

s = = = syndrome sequence (see p. 47)

T = = = syndrome threshold (see p. 151)

t = = = error-correcting capability (see p. 85)

T_o = = = optimum threshold (see p. 151)

u = = = message (or information) sequence (see p. 25)

v = = = channel sequence (see p. 25) w[?] = = Hamming weight of ?

¥ = = = ••••• (see p. 89)

θ = = = theta function (see p. 218)

$\lfloor x \rfloor$ = = greatest integer $\le x$

$\lceil x \rceil$ = = smallest integer $\ge x$

$\equiv$ = = = congruence symbol (see p. 441)

$\hat{=}$ = = = equal by definition

<A,B> = partitioned by sets A & B (see p. 352)

$A \subseteq B$ = = A is a subset of B

$A \subset B$ = = A is a proper subset of B

(a,b) = greatest common divisor of a & b (see p. 435)

x/y/z = (x/y)/z = x/(yz)

VOLUME 2

APPENDIX 1.1: THE FUNCTIONAL BLOCK-UNITS OF A DIGITAL COMMUNICATIONS SYSTEM

In this appendix, the task of each of the functional block-units of the digital communications system of Fig. 1.1 (p. 2) will be briefly described. The idea of such a diagram was borrowed from Sklar [1], from whom some of the following material is also taken.

The *information source* is either the human or the machine that originates the information to be transmitted. The information may be an *analogue signal* (i.e. a signal continuous both in amplitude and in time), or a *sampled signal* (i.e. a signal continuous in amplitude but discrete in time), or a *digital signal* (i.e. a signal discrete both in amplitude and in time).

The *information sink*, or *destination* is the human or machine that will receive an estimate of the original information signal. The signal should be delivered in a format suitable to the particular destination. The performance of the whole system is judged by the quality of the delivered signal (an ideal system would deliver an estimate which is identical to the original signal), by the delay involved, by the cost of transmission (or storage) and, for some applications, by the security against interception.

The *source formatting* unit converts the source signal into a bit stream (since the system is digital). For example, the source formatting unit for an analogue source (audio, etc) may be a PCM encoder, while for a digital source (computer terminal, etc) an ASCII (or similar) encoder (in the case of the computer terminal this is incorporated in the keyboard).

The *destination formatting* unit converts the received bit-stream into a signal suitable to the particular destination. In the case of audio signals the formatting device may be a PCM decoder, while in the case of digital signals it may be the appropriate part of a VDU or a printer.

The *source encoder* compresses the information signal. The ratio of the bit rate out of the encoder over the bit rate in the encoder is called the *compression ratio*. Compression techniques for analogue sources include differential PCM, adaptive delta modulation and linear predictive quantization; these are both source-formatting and source-coding techniques. Digital sources are compressed by variable-length coding techniques, like the Huffman and Liv-Zempel ones. The latter algorithm is adaptive in the sense that it requires no prior knowledge of the source statistics.

The *source decoder* performs the reverse operation. It assumes that no errors have occured. The validity of this assumption depends on the particular system. Usually, a single bit in error may appear with probability less than, say, 10^{-8}. In most of the cases the source decoders are able to recover*, in which case the user experiences a short or long burst of erroneous data.

Encryption prevents unauthorized users from extracting information from the channel (*privacy*) and from injecting information into the channel (*authentication*). The message is encrypted with an invertible transformation, to produce the ciphertext which is then transmitted over a public channel.

Decryption is equivalent to inverting the original transformation. This is easily done if a specific transformation-parameter is available. This parameter is called the *key* and is not available to the unauthorized user (*cryptanalyst*). The latter is assumed to have full knowledge of the transformation used and of the ciphertext, to have access to the best (specialized or not) computer systems, but not to have the key. The security of the system is based on the vast number of calculations required to decipher the ciphertext, without the key.

Channel coding aims at offering a flexibility to the system-designer to 'play' with the error-rate performance, the power requirement or the bandwidth requirement. So, for a

* In some applications, this may not be desirable.

given input data-rate one of these three parameters can be improved, at the expense of the other two. This is achieved by introducing controlled redundancy, into the encoder input-stream, which for this purpose is broken into blocks of k; the introduction of redundancy results in an increased bit-rate at the O/P of the encoder (for every k bits, n bits are transmitted by the encoder, where k<n).

The *channel decoder* uses the received bit-stream to either detect the presence of errors (and ask for a retransmission) or to correct them. Error detection & retransmission is called *automatic request for retransmission* (ARQ), while error correction is called *forward error-control* (FEC). Note that ARQ results in a variable throughput, but it is expected to offer superior error-performance.

Multiplexing (MUX) is the sharing of a communications resource (CR). Mux of bit streams is achieved by sharing the CR in time (*time-division mux* - TDM). TDM may be static (as used in telephony) or dynamic (usually called *statistical mux*). Another very common type of mux is frequency-division mux (FDM), but this operates on waveforms, hence it would be located somewhere after the modulator.

Demultiplexing separates the multiplexed bit-stream into its constituent parts.

The *modulator* is the interface between the bit-stream and the waveform parts of the system. The modulator-demodulator pair is the most essential part of the whole system. The modulator superimposes the bit-stream onto a carrier (usually a sine-wave). This is done because the frequency characteristics of an, appropriately designed, (modulated) carrier better match the channel characteristics. A sine-wave is completely defined by its three parameters, amplitude, frequency and phase. In *amplitude modulation* (AM) the carrier amplitude is made to vary in sympathy with the message signal; in *frequency modulation* (FM) the parameter which is altered is the (instantaneous) frequency of the carrier; in *phase modulation* (PM) it is the carrier phase that changes in sympathy with the message signal. If the message signal is digital (as in Fig. 1.1), these three techniques are called ASK, FSK & PSK, respectively (the initials "SK" stand for *shift-keying*). A hybrid combination of ASK & PSK is called *quadrature amplitude modulation* (QAM). In essence, a modulator maps blocks of k bits into an alphabet of 2^k waveforms.

Demodulation is the process of extracting the message signal from the received modulated carrier. The received signal may be demodulated in a coherent or noncoherent way. A *coherent demodulator* multiplies and integrates (*correlates*) the received signal with each of the prototype waveforms and chooses the one which better satisfies a certain criterion (usually the minimum Euclidean distance). For this processing to be successful, the demodulator must have knowledge of the carrier's phase reference. Noncoherent demodulators do not require knowledge of the carrier phase; this results in simpler implementation (there is no need for carrier tracking), but in worse error-rate performance.

Multiple access is, like mux, a CR sharing technique. The two differ in that multiple access usually involves the remote accessing of a resource and a guard-time overhead (required to make the controller aware of the user's demand). In MUX the CR controller has instantaneous knowledge of all the users demands.

The *transmitter* (XMT) includes a power amplifier, a frequency-up conversion stage (optional) and an antenna (or, in general, XMT-to-channel interface).

The *receiver* (RCV) includes an antenna (or, in general, a channel-to-RCV interface), a front-end amplifier and a frequency-down conversion stage (optional).

Synchronization (SYNC) is the alignement of the time scales of spatially separated time-processes. Bit SYNC is involved with the extraction of a clock signal, at the pulse-repetition frequency. Frame SYNC is involved with the detection of frame-timing slips and the recovery from such slips. Carrier SYNC is involved with the extraction of carrier phase information.

Channel is the medium between the XMT and the RCV antennae (or equivalent).

APPENDIX 1.2: BINARY PSK WITH COHERENT DEMODULATION

A1.2.1. PSK Modulation

Consider a PSK modulator with output alphabet $\{s_0(t), s_1(t)\}$, where

$$s_0(t) = \sqrt{(2E/T)}\sin(2\pi f_o t+\pi/2) \quad /0\le t\le T \qquad \text{(A1.2.1a)}$$

$$s_1(t) = \sqrt{(2E/T)}\sin(2\pi f_o t-\pi/2) \quad /0\le t\le T \qquad \text{(A1.2.1b)}$$

$$\text{and } f_o T = \text{integer} \qquad \text{(A1.2.1c)}$$

From the identity $\sin(a+b) = \sin(a)\cos(b)+\sin(b)\cos(a)$ and eqns (A1.2.1):

$$-s_1(t) = s_0(t) = \sqrt{(2E/T)}\cos(2\pi f_o t) \quad /0\le t\le T \qquad \text{(A1.2.2)}$$

The modulation rate is 1/T baud and, since transmission is binary, the data signalling rate is 1/T bps.

The energy of $s_0(t)$, or $s_1(t)$, in the time interval [0,T] is

$$\underline{\text{Energy}} = \int_0^T s_1^2(t)dt$$

$$= (2E/T)\int_0^T \cos^2(2\pi f_o t)dt \qquad \text{[using (A1.2.2)]}$$

$$= (2E/T)\int_0^T \{[1+\cos(4\pi f_o t)]/2\}dt \qquad *$$

$$= (E/T)\int_0^T dt + (E/T)\int_0^{4\pi f_o} \cos x dx/(4\pi f_o)$$

$$= E + [E/4\pi f_o T)][\sin(4\pi f_o T)-\sin 0]$$

$$= E \qquad \text{[since } f_o T=\text{integer, by (A1.2.1c)]}$$

Hence,

* Use was made of the identity $\cos^2 x = [1+\cos(2x)]/2$.

$$\underline{\text{Energy/bit}} = \int_0^T s_0^2(t)dt = \int_0^T s_1^2(t)dt = E \qquad (A1.2.3)$$

Then,

$$\text{Power} = E/T \qquad (A1.2.4)$$

Note also that, by (A1.2.3):

$$\int_0^T s_0(t)s_1(t)dt = -\int_0^T s_0^2(t)dt = -E \qquad (A1.2.5)$$

Hence, $s_0(t)$ & $s_1(t)$ are not orthogonal.

A1.2.2. The Output of a PSK Coherent Demodulator

A coherent demodulator for PSK multiplies the received signal, $r(t)$, by $s_0(t)$ [or $s_1(t) = -s_0(t)$], integrates the product from time $t=0$ and samples the integrator's O/P at time $t=T$ (see Fig. A1.2.1).

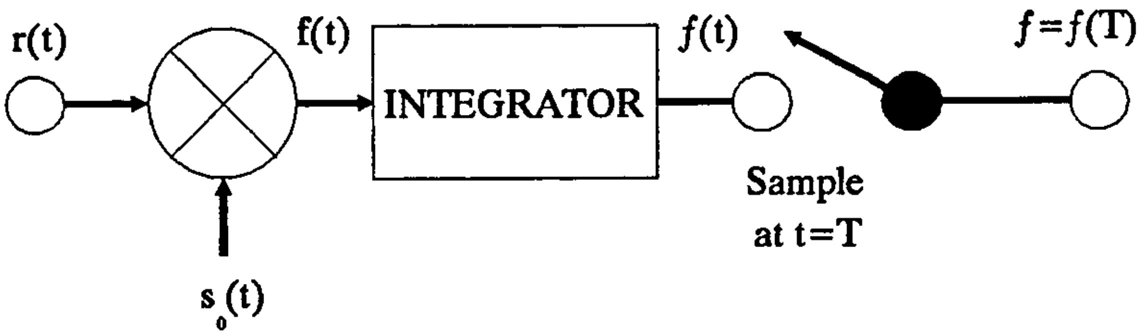


Figure A1.2.1: Coherent demodulator for binary PSK.

Assuming a channel suffering from additive noise, $n(t)$,

$$r(t) = s_i(t)+n(t) \quad /i = 0 \text{ or } 1 \qquad (A1.2.6)$$

From Fig. A1.2.1 and eqn (A1.2.6):

$$f(t) = r(t)s_0(t) = s_i(t)s_0(t) + n(t)s_0(t) \Longrightarrow$$

$$f(t) = \pm s_0^2(t) + n(t)s_0(t) \Longrightarrow$$

$$f(t) = \int_0^t [\pm s_0^2(x) + n(x)s_0(x)]dx \Longrightarrow$$

$$f(T) \hat{=} f = \pm\int_0^T s_0^2(t)dt + \int_0^T n(t)s_0(t)dt$$

[and using (A1.2.3)], $$f = \pm E+n_c \qquad (A1.2.7)$$

where $$n_c \hat{=} \int_0^T n(t)s_0(t)dt \qquad (A1.2.8)$$

Note that n_c is a random variable because the waveform $n(t)$ $/0 \le t \le T$ is random. Hence the demodulator O/P is $E+n_c$ if $s_0(t)$ was transmitted, or $-E+n_c$ if $s_1(t)$ was transmitted. Note also that f is a real random variable, which has to be further processed in order to be determined whether $s_0(t)$, or $s_1(t)$, was transmitted. Of course such a decision will not always be error-free, due to the random nature of n_c.

In order to determine the optimum way to further process f, one has to examine the statistical properties of n_c.

A1.2.3. Statistical Properties of n_c

If the additive noise $n(t)$ is Gaussian, then $n(t)$ is a Gaussian stochastic process, hence $n(t')$ is a Gaussian random variable. Since $s_0(t')$ is a constant, then $n(t')s_0(t')\delta t$ is also a Gaussian random variable (with different mean and variance - see Papoulis [3], p. 127), where δt is a small time interval.

Consider the sum,

$$\sum_{i=1}^{T/\delta t} n[\delta t/2+\delta t(i-1)]s_0[\delta t/2+\delta t(i-1)]\delta t \qquad (A)$$

Let the additive Gaussian noise be also white, with power spectral density (double-sided) $\tilde{n}/2$ (this means that the noise power from $f=f_0$ to $f=f_0+B$ is $2B\tilde{n}/2 = \tilde{n}B$). Since the power spectral density, $G_n(f)$, is constant over all frequencies*, the autocorrelation function, $R_n(\tau)$, of the Gaussian white noise process is impulsive, because $R_n(\tau)$ & $G_n(f)$ form a Fourier-transform pair (see Papoulis [3], p. 338):

$$R_n(\tau) = (\tilde{n}/2)\delta(\tau) \qquad (A1.2.9)$$

This means that $E[n(t_1)n(t_1+\delta t)] = 0$, hence the factors in summation (A) are statistically independent Gaussian ran-

* In practice, up to about 10^{13} Hz.

dom variables. Then, their sum is also a Gaussian random variable (see Davenport [4], pp. 188-90). If one lets $\delta t \longrightarrow 0$, the summation in (A) tends to the integral, in (A1.2.8), which defines n_c. Hence, <u>n_c is a Gaussian random variable</u>.

$E[n_c]$, the <u>expected value</u> of n_c, may be obtained from (A1.2.8):

$$E[n_c] = E\left[\int_0^T n(t)s_0(t)dt \right] \qquad \text{(B)}$$

From (B), $E[n_c]$ is the ensemble average over the noise voltages, $n(t)$, from all statistically independent noise sources. If $n_j(t)$ denotes a noise sample-function from the jth source, the RHS of (B) may be written as (N is the number of noise sources),

$$\lim_{N \to +\infty} (1/N)\sum_{j=1}^{N}\left[\int_0^T n_j(t)s_0(t)dt \right]$$

and using (A),

$$E[n_c] = \lim_{\substack{\delta t \to 0 \\ N \to +\infty}} (1/N)\left[\sum_{j=1}^{N} \sum_{i=1}^{T/\delta t} n_j(t_i)s_0(t_i)\delta t \right] \longrightarrow$$

$$E[n_c] = \lim_{\delta t \to 0} \sum_{i=1}^{T/\delta t} \left[\lim_{N \to +\infty}(1/N) \sum_{j=1}^{N} n_j(t_i) \right] s_0(t_i)\delta t \qquad \text{(C)}$$

The summation in the brackets in (C), is the ensemble average of noise samples at time $t=t_i$, over all noise sources. Then:

$$\lim_{N \to +\infty} (1/N)\left[\sum_{j=1}^{N} n_j(t_i) \right] = E[n(t_i)] \qquad \text{(D)}$$

Since the Gaussian noise process is stationary, the statistical averages are independent of time and $E[n(t_i)] = E[n(t)] = 0$. Then:

$$E[n_c] = 0 \qquad \text{(A1.2.10)}$$

Finally, consider the <u>mean square value</u>, $E[n_c^2]$, of n_c. From (A1.2.8):

$$n_c^2 = \int_0^T n(t)s_0(t)dt \int_0^T n(x)s_0(x)dx \implies$$

$$E[n_c^2] = E\left[\int_0^T\int_0^T n(t)n(x)s_0(t)s_0(x)dtdx\right]$$

and since the operations of integration & expectation are interchangeable *,

$$E[n_c^2] = \int_0^T\int_0^T E[n(t)n(x)s_0(t)s_0(x)]dtdx$$

Since, for ensemble averaging, $s_0(t)$ & $s_0(x)$ are constants (the ensemble average is over all noise sample-functions, at a fixed time):

$$E[n_c^2] = \int_0^T\int_0^T E[n(t)n(x)]s_0(t)s_0(x)dtdx \qquad (E)$$

$E[n(t)n(x)]$ is the autocorrelation function $R_n(t,x)$ of the noise process, which is a function of only the difference t-x, because the process is stationary (see Davenport [4], pp. 322-3). Hence, from (A1.2.9):

$$E[n(t)n(x)] = R_n(t-x) = (\tilde{n}/2)\delta(t-x) \qquad (F)$$

From (E) & (F):

$$E[n_c^2] = \int_0^T\int_0^T R_n(t-x)s_0(t)s_0(x)dtdx \implies$$

$$E[n_c^2] = (\tilde{n}/2)\int_0^T s_0(x)\left[\int_0^T s_0(t)\delta(t-x)dt\right]dx \qquad (G)$$

If f(x) is any function, continuous at the origin, then the *shifting property* of *Dirac's delta function* is (see Papoulis [3], p. 97):

$$\int_{-\infty}^{+\infty} f(x)\delta(x-\alpha)dx = f(\alpha)$$

Since x ranges from 0 to T, the value t=x is within the range of variation of t, hence the range $(-\infty,+\infty)$ may be replaced by [0,T]:

$$\int_0^T s_0(t)\delta(t-x)dt = s_0(x) \qquad (H)$$

* See Papoulis [3], Chapter 9; see also the argument leading to result (A1.2.10).

From (G), (H) & (A1.2.3):

$$E[n_c^2] = \tfrac{1}{2}\tilde{n}\int_0^T s_0^2(x)dx = \tfrac{1}{2}\tilde{n}E \qquad (A1.2.11)$$

A1.2.4. Hard-Decision Demodulation

The coherent demodulator's output is $\pm E+n_c$, where n_c is a zero-mean Gaussian random variable with variance $E[n_c^2] = \sigma^2 = E\tilde{n}/2$. The hard-decision threshold, T, is set at 0. *

Hence, if $s_0(t)$ is transmitted, $E+n_c$ is received, while if $s_1(t)$ is transmitted $-E+n_c$ is received. Then, the optimum hard-decision demodulation rule is:

$$f = \begin{cases} \rightarrow s_0(t) \text{ if } f \geq 0 \\ \rightarrow s_1(t) \text{ if } f < 0 \end{cases} \qquad (A1.2.12)$$

A1.2.5. Probability of Error

A demodulation error will occur if n_c exceeds certain limits. Specifically,

Error, if $s_0(t)$ is transmitted and $f<0$,
OR, if $s_1(t)$ is transmitted and $f\geq 0$.

$$\Longleftrightarrow$$

Error, if $s_0(t)$ is transmitted and $n_c<-E$,
OR, if $s_1(t)$ is transmitted and $n_c\geq +E$. (A1.2.13)

From (A1.2.13),

$$P_e = P[s_0(t)]P(n_c<-E) + P[s_1(t)]P(n_c\geq +E)$$

Since $P(n_c\geq +E) = P(n_c<-E)$,

$$P_e = P(n_c\geq +E) \qquad (A1.2.14)$$

Since n_c is a zero-mean Gaussian random variable with variance σ^2, then:

$$P_e \triangleq P(n_c \geq E) = \int_E^{+\infty}[e^{-x^2/(2\sigma^2)}/\sqrt{(2\pi\sigma^2)}]dx$$

* If the bit error rate, P_e, is expressed as $P_e = P(s_0)P(f<T)+P(s_1)P(f>T) = P(s_0)P(n_c<T-E) +[1-P(s_0)]P(n_c>T+E)$, differentiated with respect to T and set equal to 0, it is found that T, the *optimum threshold*, is 0.

Let $x/(\sigma\surd 2) = z$. Then,

$$P_e = 1/\surd(2\pi\sigma^2)\int_{E/(\sigma\surd 2)}^{+\infty} e^{-z^2}\sigma\surd 2dz = \tfrac{1}{2}(2/\surd\pi)\int_{E/(\sigma\surd 2)}^{+\infty} e^{-z^2}dz \longrightarrow$$

$$P_e = \tfrac{1}{2}\mathrm{erfc}[E/(\sigma\surd 2)] \qquad (A1.2.15)$$

where $$\mathrm{erfc}(z) \mathrel{\hat{=}} (2/\surd\pi)\int_z^{+\infty} e^{-x^2}dx \qquad (A1.2.16)$$

Using (A1.2.11) in (A1.2.15):

$$P_e = \tfrac{1}{2}\mathrm{erfc}[\surd(E/\tilde{n})] \qquad (A1.2.17)$$

A1.2.6. Bounds on, and Approximation to, P_e

The following theorem was taken from Feller [5] (p. 175):

Theorem A1.2.1: If $Q(x)$ is the complementary normal (Gaussian) distribution, defined by

$$Q(x) \mathrel{\hat{=}} \int_x^{+\infty} f(z)dz \qquad (A1.2.18)$$

where $f(z)$ is the, zero-mean, unit-variance, normal density-function, given by

$$f(z) = e^{-z^2/2}/\surd(2\pi) \qquad (A1.2.19)$$

then:

$$(1-1/x^2)f(x)/x < Q(x) < f(x)/x \qquad (A1.2.20)$$

$$Q(x) \approx f(x)/x \quad \text{as } x \longrightarrow +\infty \qquad (A1.2.21)$$

Proof: For all $z>0$:

$$1-3/z^4 < 1 < 1+1/z^2 \qquad (A)$$

Since, $f(z)>0$, from (A):

$$(1-3/z^4)f(z) < f(z) < (1+1/z^2)f(z) \qquad (B)$$

From (B) and (A1.2.18), for all $x>0$:

$$\int_x^{+\infty}(1-3/z^4)f(z)dz < \int_x^{+\infty}f(z)dz < \int_x^{+\infty}(1+1/z^2)f(z)dz \longrightarrow$$

$$I_1(x) < Q(x) < I_2(x) \qquad (C)$$

where $$I_1(x) \triangleq \int_x^{+\infty}(1-3/z^4)f(z)dz \qquad (D)$$

and $$I_2(x) \triangleq \int_x^{+\infty}(1+1/z^2)f(z)dz \qquad (E)$$

From (A1.2.19): $$df(z)/dz = -zf(z) \qquad (A1.2.22)$$

From (E), (A1.2.18) & (A1.2.22):

$$I_2(x) = \int_x^{+\infty}f(z)dz + \int_x^{+\infty}[f(z)/z^2]dz \quad \Longrightarrow$$

$$I_2(x) = Q(x) + \int_x^{+\infty}f(z)d(-1/z) \quad \Longrightarrow$$

$$I_2(x) = Q(x) - \Big[(1/z)f(z)\Big]_x^{+\infty} + \int_x^{+\infty}(1/z)df(z) \quad \Longrightarrow$$

$$I_2(x) = Q(x) - [0-f(x)/x] - \int_x^{+\infty}f(z)dz \quad \Longrightarrow$$

$$I_2(x) = f(x)/x \qquad (F)$$

Also, from above,

$$\int_x^{+\infty}[f(z)/z^2]dz = f(x)/x - Q(x) \qquad (G)$$

From (D), (A1.2.18), (A1.2.22) & (G):

$$I_1(x) = \int_x^{+\infty}f(z)dz - 3\int_x^{+\infty}[f(z)/z^4]dz \quad \Longrightarrow$$

$$I_1(x) = Q(x) - 3\int_x^{+\infty}f(z)d[-1/(3z^3)] \quad \Longrightarrow$$

$$I_1(x) = Q(x) - 3\Big[-f(z)/(3z^3)\Big]_x^{+\infty} + 3\int_x^{+\infty}[-1/(3z^3)]df(z) \quad \Longrightarrow$$

$$I_1(x) = Q(x) + \Big[f(z)/z^3\Big]_x^{+\infty} + \int_x^{+\infty}[f(z)/z^2]dz \quad \Longrightarrow$$

$$I_1(x) = Q(x) + [0-f(x)/x^3] + f(x)/x - Q(x) \quad \Longrightarrow$$

$$I_1(x) = (1-1/x^2)f(x)/x \qquad (H)$$

From (H), (F) & (C), (A1.2.20) follows readily.

Note that the difference between the upper and the lower bound on $Q(x)$ [from (A1.2.20)] is $1/x^2$, which tends to 0, as $x \longrightarrow +\infty$. Then, $Q(x) \approx f(x)/x$, as $x \longrightarrow +\infty$.

QED

The results of Theorem A1.2.1, will be used now to obtain bounds on, and an approximation to, P_e.

Lemma A1.2.1: The probability of a bit-error, for binary PSK transmission over the AWGN channel and coherent demodulation with hard-decisions, is bounded by

$$[1-1/(2\Gamma)]e^{-\Gamma}/[2\sqrt{(\pi\Gamma)}] < P_e < e^{-\Gamma}/[2\sqrt{(\pi\Gamma)}] \qquad \text{(A1.2.23)}$$

$$\text{where} \quad \Gamma \triangleq E/\tilde{n} \qquad \text{(A1.2.24)}$$

E is the energy per received bit and $\tilde{n}/2$ is the double-sided noise power spectral density.

Furthermore:

$$P_e \approx e^{-\Gamma}/[2\sqrt{(\pi\Gamma)}] \quad \text{as} \quad \Gamma \longrightarrow +\infty \qquad \text{(A1.2.25)}$$

Proof: From the definition of $Q(x)$ & $\mathrm{erfc}(x)$ [see (A1.2.18) & (A1.2.16)], the following relationship is obtained:

$$Q(x) \triangleq [1/\sqrt{(2\pi)}]\int_x^{+\infty} e^{-z^2/2}dz \qquad (\text{let } y=z/\sqrt{2})$$

$$= \tfrac{1}{2}[2/\sqrt{(2\pi)}]\int_{x/\sqrt{2}}^{+\infty} e^{-y^2}\sqrt{2}dy$$

$$= \tfrac{1}{2}(2/\sqrt{\pi})\int_{x/\sqrt{2}}^{+\infty} e^{-y^2}dy = \tfrac{1}{2}\mathrm{erfc}(x/\sqrt{2}) \quad \longrightarrow$$

$$Q(x) = \tfrac{1}{2}\mathrm{erfc}(x/\sqrt{2}) \qquad \text{(A1.2.26)}$$

From (A1.2.26) & (A1.2.20):

$$(1-1/x^2)f(x)/x < \tfrac{1}{2}\mathrm{erfc}(x/\sqrt{2}) < f(x)/x \quad \longrightarrow$$

$$[1-1/(y\sqrt{2})^2)]f(y\sqrt{2})/(y\sqrt{2}) < \tfrac{1}{2}\mathrm{erfc}(y) < f(y\sqrt{2})/(y\sqrt{2}) \quad \longrightarrow$$

$$[1-1/(2y^2)]e^{-y^2}/[y\sqrt{2}\sqrt{(2\pi)}] < \tfrac{1}{2}\mathrm{erfc}(y) < e^{-y^2}/[y\sqrt{2}\sqrt{(2\pi)}]$$

[from (A1.2.19) - let now $y=\sqrt{\Gamma}$]

$$\longrightarrow \quad [1-1/(2\Gamma)]e^{-\Gamma}/[2\sqrt{(\pi\Gamma)}] < \tfrac{1}{2}\mathrm{erfc}(\sqrt{\Gamma}) < e^{-\Gamma}/[2\sqrt{(\pi\Gamma)}]$$

Approximation (A1.2.25) & bounds (A1.2.23) follow readily.

QED

APPENDIX 1.3: AVERAGE ERROR-RATE FOR A SIMPLE CHANNEL WITH MEMORY

If P(b) = 1-P(g) is the probability that the channel will be found in the 'bad' state, then from Fig. 1.4, the probability, P(g), that the channel will be found in the 'good' state is:

$$P(g) = P(g)(1-q_1)+P(b)q_2 = P(g)(1-q_1)+[1-P(g)]q_2 \Longrightarrow$$

$$P(g)(1-1+q_1+q_2) = q_2 \Longrightarrow P(g) = q_2/(q_1+q_2) \qquad (A1.3.1)$$

Since P(b)=1-P(g),

$$P(b) = q_1/(q_1+q_2) \qquad (A1.3.2)$$

Note that since $q_1 \ll q_2$, $P(g) \approx 1$ & $P(b) \approx q_1/q_2$.

The average probability of error for this channel is

$$P_E = P(g)p_1+P(b)p_2 \Longrightarrow$$

$$\Longrightarrow P_E = (q_2p_1+q_1p_2)/(q_1+q_2) \approx p_1+(q_1/q_2)p_2 \qquad (A1.3.3)$$

APPENDIX 1.4: ASYMPTOTIC CODING GAIN FOR A BLOCK CODE

Consider the calculation of the asymptotic coding-gain for a t-error correcting code, of rate R, with BPSK transmission over the AWGN channel and coherent demodulation with hard decisions.

Let p denote the probability of a bit in error over the BSC (made of the BPSK modulator, the AWGN and the coherent demodulator). From eqn (1.7), the probability of erroneous decoding, P(E), is

$$P(E) = \sum_{i=n\beta+1}^{n} p^i(1-p)^{n-i}C(n,i)$$

Since at high SNRs, p is very small, only the first term of the above summation is significant*. Also $(1-p)^{n-t-1} \approx 1$, so $P(E) \approx Kp^{t+1}$, where K is a constant. For the same reason as above, given that a block is erroneously decoded [and that happens with probability P(E)], the probability that it contains more than t+1 errors is close to zero (because p is very small at high SNRs). Hence, P(E) is approximately equal to the probability of t+1 errors. So, the bit error-rate, P_b, at the decoder O/P is:

$$P_b \approx K(t+1)p^{t+1} \qquad (A1.4.1)$$

p is the channel error-rate as 'seen' by the decoder. Hence, p is given by (1.4), but the SNR per information-bit, Γ, is reduced by a factor of R:

$$P_b \approx K(t+1)\{\tfrac{1}{2}\mathrm{erfc}[\surd(\Gamma R)]\}^{t+1} \qquad (A1.4.2)$$

For uncoded transmission, the bit error-rate, P_b', is

$$P_b' = \tfrac{1}{2}\mathrm{erfc}(\surd\Gamma') \qquad (A1.4.3)$$

The expressions in eqns (A1.4.2) & (A1.4.3), for high SNRs, can be approximated by (1.5c). Then, to achieve the same bit error-rate ($P_b'=P_b$):

$$K(t+1)\{e^{(-\Gamma R)}/[2\surd(\pi\Gamma R)]\}^{t+1} = e^{-\Gamma'}/[2\surd(\pi\Gamma')] \implies$$

$$\ln[K(t+1)] + (t+1)\{-\Gamma R-\ln[2\surd(\pi\Gamma R)]\} = -\Gamma'-\ln[2\surd(\pi\Gamma')] \implies$$

$$\Gamma' = R\Gamma(t+1)+(t+1)\ln[2\surd(\pi\Gamma R)]-\ln[2\surd(\pi\Gamma')]-\ln[K(t+1)] \implies$$

$$\Gamma' = R\Gamma(t+1)+\tfrac{1}{2}\ln[(\Gamma R)^{t+1}/\Gamma']+t\ln(2\surd\pi)-\ln[K(t+1)]$$

Since all logarithmic factors are small, for $\Gamma \to \infty$:

$$\Gamma'/\Gamma \approx R(t+1) \implies G_a \approx 10\log[R(t+1)] \qquad (A1.4.4)$$

* This is the i=t+1 term, since $n\beta = t$.

APPENDIX 2.1: INTRODUCTION TO ABSTRACT ALGEBRA

This appendix is intended to serve as a 'look-up table' for basic definitions and theorems of abstract algebra.

The part of convolutional-code theory, covered by this thesis, is inherently algebraic. Consequently, the reader is expected to be familiar with the most common elements of abstract algebra.

More information can be found in chapter 2 of most textbooks on error-correcting codes.

Definition A2.1.1: A set S, together with an operation $*$ defined in the elements of S, forms a *group* G if the following properties are satisfied:

i) *Closure*: For every a,b in S, $a*b$ is in S.

ii) *Associativity*: For every a,b,c in S, $a*(b*c) = (a*b)*c$.

iii) *Identity*: S contains an element e such that, for all b in S, $b*e = b$.

iv) *Inverse*: For every b in S there is an element c, in S, such that $b*c = e$. c is called the inverse of b and is denoted by b^{-1}.

∎

Theorem A2.1.1: In every group, the identity element is unique. Also, the inverse of each group element is unique, and $(a^{-1})^{-1} = a$.

∎

Definition A2.1.2: If a group G satisfies the commutative property, i.e. if for every a,b in G, $a*b = b*a$, then the group is called *commutative* or *abelian*.

∎

Definition A2.1.3: If a group G has a finite number of elements, it is called a *finite group* and the number of elements in G is called the *order* of G.

∎

Definition A2.1.4: A set *S*, together with an operation * defined on the elements of *S*, forms a *semigroup* if * is a closed associative operation.

∎

Definition A2.1.5: A set *S* together with two operations on *S*, addition (denoted by +) and multiplication (denoted by juxtaposition), forms a *ring* if:

i) *S* together with addition forms an abelian group.
ii) *S* together with multiplication forms a semigroup.
iii) The *distributive laws* a(b+c) = ab+ac and (b+c)a = ba+ca, hold.

∎

Definition A2.1.6: A set *S* together with two operations on *S*, addition and multiplication, forms a *field* if:

i) *S* together with addition forms an abelian group with additive identity denoted by 0.
ii) $S' = \{s : s \in S \ \& \ s \neq 0\}$ together with multiplication forms an abelian group.
iii) The *distributive law* a(b+c) = ab+ac holds for all a,b,c in *S*.

∎

Definition A2.1.7: A field with q elements, if it exists, is called a *finite field*, or *Galois field*, and is denoted by GF(q).

∎

Definition A2.1.8: Let *F* be a field. The elements of *F* will be called *scalars*. A set *V* is called a *vector space* and its elements are called *vectors* if there is defined an operation called vector addition (denoted by +) on pairs of elements from *V*, and an operation called scalar multiplication (denoted by juxtaposition) on an element from *F* and an element from *V*, provided the following hold true:

i) *V* is an abelian group under vector addition.
ii) *Distributive law*: For any vectors v_1 & v_2 and any scalar c, $c(v_1+v_2) = cv_1+cv_2$.

iii) *Distributive law*: For any vector v and any scalars c_1 & c_2, $(c_1+c_2)v = c_1v+c_2v$.

iv) *Associative law*: For any vector v and any scalars c_1 & c_2: $(c_1c_2)v = c_1(c_2v)$.

v) If 1 is the multiplicative identity of *F*, $1v = v$, for all v in *V*.

∎

<u>Definition A2.1.9</u>: Let *S* be an non-empty subset of a vector space *V*. *S* is a *vector subspace* if it forms a vector space under the original vector addition and scalar multiplication.

∎

<u>Definition A2.1.10</u>: In a vector space *V*, a sum of the form

$$u = a_1v_1+a_2v_2+\cdots+a_kv_k$$

where the a_i are scalars, is called a *linear combination* of the vectors $v_1,v_2,\ldots,v_k$.

A set of vectors $\{v_1,v_2,\ldots,v_k\}$ is called *linearly dependant* if there exist scalars $a_1,a_2,\ldots,a_k$ such that

$$a_1v_1+a_2v_2+\cdots+a_kv_k = 0$$

∎

<u>Definition A2.1.11</u>: If $a_i \in F$ /$i=1,2,\ldots,k$, where *F* is a field, the quantity $(a_1,a_2,\ldots,a_k)$ is called a *k-tuple* of elements from the field *F*. Under the operations of componentwise addition and componentwise scalar multiplication, the set of k-tuples of elements from a field *F* forms a vector space over *F*, which is denoted by F^k.

∎

<u>Definition A2.1.12</u>: A set of vectors is said to *span* a vector space if every vector in the space equals at least one linear combination of the vectors in the set. A vector space that is spanned by a finite set of vectors is called a *finite-dimensional vector space*.

∎

<u>Definition A2.1.13</u>: The number of vectors in a set that spans a finite-dimensional vector space *V* is called the *di-*

mension of *V*. A set of k linearly independent vectors that span a k-dimensional vector space *V* is said to form a *basis* of *V*.

∎

Note A2.1.1: Any finite-dimensional vector space *V* can be represented as an n-tuple space: If the set of vectors $\{v_1, v_2, \ldots, v_n\}$ forms a basis of *V* then every $v \in V$ can be expressed as $v = a_1 v_1 + a_2 v_2 + \cdots + a_n v_n$, hence one may represent v by the n-tuple of coefficients $(a_1\ a_2 \cdots a_n) = v$.

∎

Definition A2.1.14: A *single-valued mapping* of a set *S into a set T* is a correspondence ($f: s \longrightarrow sf$) that associates with each $s \in S$ a unique element $t \in T$. Two mappings f & g, of *S* into *T*, are equal ($f = g$) iff $sf = sg$ for all $s \in S$. A mapping of *S* into *T* is a *mapping of S onto T*, if for each $t \in T$ there exists at least one $s \in S$: $sf = t$. f is a *one-to-one mapping* iff for each $a, b \in S$: $a \neq b \Longrightarrow af \neq bf$. [6]

∎

Definition A2.1.15: Let *S* & *T* be any two sets. The set $S \times T = \{(s,t) : s \in S, t \in T\}$ is called the *cartesian product* of the sets *S* & *T*. [6]

∎

APPENDIX 2.2: INTRODUCTION TO LINEAR ALGEBRA

This appendix is intended to give a few definitions and theorems that will be used throughout the thesis. The reader may find more information in chapters 1 & 3 of Noble & Daniel [7].

Definition A2.2.1: An m x n *matrix* over a ring *R* is made of mn elements of *R*, arranged in a rectangular array of m rows and n columns. If the elements of A are denoted by a_{ij} / $i=1,2,\ldots,m$ & $j=1,2,\ldots,n$, then the matrix can also be denoted by $A = [a_{ij}]$.

∎

<u>Definition A2.2.2</u>: The *transpose* of the m × n matrix A = $[a_{ij}]$ is the n × m matrix $A^T = [a^T_{ij}]$, such that $a^T_{ij} = a_{ji}$ /i= 1,2,...,n & j=1,2,...,m.

∎

<u>Theorem A2.2.1</u>: Properties of the transpose matrix [7]:

i) $(A+B)^T = A^T + B^T$

ii) $(A^T)^T = A$

iii) $(AB)^T = B^TA^T$

∎

<u>Definition A2.2.3</u>: A matrix G such that GA = I, if such a matrix exists, is called a *left-inverse* of A. A matrix H such that AH = I, if such a matrix exists, is called a *right- inverse* of A [7].

∎

<u>Theorem A2.2.2</u>: If both the right-inverse and the left-inverse of a matrix A exist, they are the same; this common inverse is called *the inverse* of A, is unique and is denoted by A^{-1}.

∎

<u>Theorem A2.2.3</u>: Properties of the inverse [7]:

i) A square matrix possesses an inverse or it does not posses either a left- or a right-inverse.

ii) If A & B are square matrices that posses an inverse (in which case they are called *nonsingular*):

1. $(A^{-1})^{-1} = A$

2. $(AB)^{-1} = B^{-1}A^{-1}$

3. $(A^T)^{-1} = (A^{-1})^T$

iii) The results in (ii) imply that if A & B are non-singular, so are A^T, A^{-1} & AB.

∎

<u>Definition A2.2.4</u>: *Elementary row operations* on matrices are defined as following:

i) Interchange of any two rows.

ii) Multiplication of any row by a non-zero element.

iii) Replacement of any row by the sum of itself and a multiple of any other row.

Elementary column operations are defined by replacing the term "row" by the term "column", above.[8]

∎

Definition A2.2.5: An $m \times n$ matrix is said to be *canonical* or in *row-echelon form* if:

i) Certain columns numbered $c_1 < c_2 < \cdots < c_r$ are precisely the unit vectors $e_1, e_2, \ldots, e_r$; the *unit vector* e_j, of order m $(1 \leq j \leq m)$, is the $m \times 1$ matrix with the jth element unity and all other elements zero.

ii) For a column numbered c, where $c_i \leq c < c_{i+1}$ $(1 \leq i \leq r)$, its last $m-i$ elements are zero.

From (i) & (ii) above, it follows that:

iii) The last $m-r$ rows of the canonical matrix are zero; the first r rows are non-zero.

iv) The lower triangle of elements in the (i,j) positions, where $i>j$, is all zero.

v) For row i $(1 \leq i \leq m)$:
1. The first c_i-1 elements are zero.
2. The c_ith element is 1.
3. The c_jth element is zero, for $i \neq j$.

∎

Theorem A2.2.4: Any elementary row (column) operation on an $m \times n$ matrix **A**, can also be achieved by forming the product **HA** (**AK**). **H** (**K**) is the corresponding *elementary matrix*, obtained by performing the row (column) operation on I_m (I_n). An elementary matrix is nonsingular.

∎

Definition A2.2.6: An *elementary operation* is any operation that is either an elementary row operation or an elementary column operation. If a matrix **A** can be transformed into a matrix **B** by means of one or more elementary operations, we write A ~ B and say that A is equivalent to B. In particular, we may say that A is *row equivalent* (or *column equivalent*) to B if only elementary row (or column) operations are involved in the transformation. [8]

∎

Theorem A2.2.5: The row-echelon form of a matrix is unique.

∎

Definition A2.2.7: The number of non-zero rows in the row-echelon form of a matrix is known as its *rank*.

∎

Definition A2.2.8: By means of elementary transformations any matrix A of rank r>0 can be reduced to one of the forms

$$I_r \quad , \quad [I_r, 0] \quad , \quad \begin{bmatrix} I_r \\ 0 \end{bmatrix} \quad , \quad \begin{bmatrix} I_r & 0 \\ 0 & 0 \end{bmatrix}$$

called its *normal form*. A zero matrix is its own *normal form* [9].

∎

Theorem A2.2.6: Equivalent matrices have the same rank [8].

∎

Theorem A2.2.7: Two matrices A and B are equivalent iff there exist two nonsingular matrices P and Q such that A = PBQ.

∎

Theorem A2.2.8: If A is an n x n matrix and if $[A, I_n]$ can be transformed to the equivalent matrix $[I_n, B]$ by elementary row operations, then B is the inverse of A [8].

∎

Theorem A2.2.9: Let A be a square m x m matrix of rank m. Then, [7]

i) The row-echelon form of A is I_m.

ii) A is the product of elementary matrices.
iii) A is nonsingular.

∎

<u>Theorem A2.2.10</u>: If A is a general m x m matrix and B is an m x n matrix of rank m, the rank of [B,A] is m [7].

∎

<u>Theorem A2.2.11</u>: Let A be an m x n matrix of rank r. Then, [7]

i) A has a right-inverse R $\Longleftrightarrow$ $r = m \le n$
ii) A has a left-inverse L $\Longleftrightarrow$ $r = n \le m$

∎

<u>Theorem A2.2.12</u>: Let A be a square m x m matrix. A is nonsingular iff the rank of A is m. [7]

∎

<u>Theorem A2.2.13</u>: If A & B are m x m matrices and AB is nonsingular, both A and B are nonsingular [7].

∎

<u>Theorem A2.2.14</u>: If A is nonsingular, the rank of AB (and also of BA) is that of B [9]. *

∎

<u>Theorem A2.2.15</u>: The rank of the product of two matrices cannot exceed the rank of either factor [9].

∎

<u>Theorem A2.2.16</u>: If the m x p matrix A is of rank r and the p x n matrix B is such that AB = 0, the rank of B cannot exceed p-r [9].

∎

<u>Theorem A2.2.17</u>: If A is m x n and B is n x m with n<m, AB is singular [7].

∎

<u>Theorem A2.2.18</u>: Suppose that AB = 0. Then [7]:

i) If A is n x n & B is n x p, B = 0 or A = singular.
ii) If A is m x n & B is n x n, A = 0 or B = singular.

* In fact, AB & B have the same canonical matrix.

iii) If A and B are both $n \times n$, $A = 0$, or $B = 0$, or both A & B are singular.

∎

Theorem A2.2.19: If two matrices are related by a succession of elementary row operations, they have the same row space (*row space* of a matrix is the set of all linear combinations of its rows).

∎

Theorem A2.2.20: Let A be an $m \times n$ matrix with elements in GF(q). The *row space* of A is a vector sub-space of $GF(q)^n$, with dimension equal to the rank of the matrix. The *column space* of A, the set of all linear combinations of the columns of A, is a vector subspace of $GF(q)^m$ with dimension equal to the rank of A.

∎

Theorem A2.2.21: Let A be an $m \times n$ matrix with elements in GF(q). The set of n-tuples v such that $Av^T = 0$ is called the *null-space* of A and forms a vector subspace of $GF(q)^m$.

∎

NOTE: Information about the proof of the theorems of Appendix 2.2, can be found in Appendix 2.3.

APPENDIX 2.3: PROOF OF THE THEOREMS IN APPENDIX 2.2

This appendix is intended to provide the reader with a brief sketch of the proofs of those theorems of Appendix 2.2, for which a reference was not found.

For Theorems A2.2.1, A2.2.2 & A2.2.3: See Noble & Daniel [7], pp. 11-8.

∎

For Definition A2.2.5, parts (iii), (iv) & (v): Using parts (i) & (ii):

iii) If $c_i \leq c < c_{i+1}$ $(1 \leq i \leq r)$, the column numbered c has its last m-i elements zero. Then the last MIN{m-i} elements of each column are zero, hence the last m-MAX{i} = m-r rows are zero.

iv) The elements of the lower triangle are $\alpha(p,c) \hat{=} \alpha_{p,c}$ with p>c. From the discussion above, if $i<p\le m$ then $\alpha(p,c)$ = 0, where $c_i \le c < c_{i+1}$ and $1\le i\le r$. Since $i\le c_i$ and $c_i\le c$, if $c<p \Longrightarrow i<p$, hence $\alpha(p,c) = 0$.

v) Column c_i contains a 1 in position i. So, $\alpha(i,c_i)$ = 1. Consider $\alpha(i,c)$ /$c<c_i$. Let $c_j\le c<c_{j+1}$ with $j+1\le i$. In column c, elements j+1,...,m are zero. Hence $\alpha(i,c) = 0$. In row i, positions $c_1,c_2,\ldots,c_r$ belong to e_j /j=1,2,...,r, respectively.

∎

For Theorem A2.2.4: See [7], pp. 85-6 for a proof for row operations. The proof for column operations is similar. The proof of the last statement is in [7], pp. 86-7.

∎

For Theorem A2.2.5: See [7], pp. 88-90.

∎

For Theorems A2.2.6, A2.2.7 & A2.2.8: See Campbell [8], pp. 130-8.

∎

For Theorem A2.2.9: Let P be the row-echelon form of A. Then P contains the r unit vectors $e_1,e_2,\ldots,e_r$ (Definition A2.2.5). Since P is m x m, of rank m, then m-r=0* and P = $[e_1,e_2,\ldots,e_m] = I_m$. So, $P = I_m = FA$ (by Theorem A2.2.4), where F is the corresponding elementary matrix. Then A has a left inverse, hence it is nonsingular (by Theorem A2.2.3). By Definition A2.2.6 & Theorem A2.2.7, $P = I_m$ & A are equivalent, hence there exist nonsingular matrices G & H such that $P = I_m = GAH \Longrightarrow A = G^{-1}H^{-1}$ (G^{-1} & H^{-1} are nonsingular, by Theorem A2.2.3). G & H are elementary, since GAH = P.

∎

For Theorem A2.2.10: Let $[P_1,P_2]$ be the row-echelon form of [B,A]. According to Definition A2.2.5, P_1 is an m x n canonical matrix, and by Theorem A2.2.4: $[P_1,P_2]$ = $F[B,A] \Longrightarrow P_1 = FB$ (F is the elementary matrix), so P_1 &

* By Definitions A2.2.5 & A2.2.7.

B are row-equivalent, hence they have the same rank (Theorem A2.2.6), so P_1 has rank m, and since $[P_1,P_2]$ is m × (m+n), it has no zero rows, hence its rank is m and so is the rank of its row-equivalent [B,A] (ibid).

∎

For Theorems A2.2.11 & A2.2.12: See [7], pp. 96-7.

∎

For Theorem A2.2.13: Since AB is nonsingular, if F is its inverse, I = F(AB) = (FA)B ⟹ FA is the left-inverse of B ⟹ the rank of B is m (Theorem A2.2.11) ⟹ B is nonsingular (Theorem A2.2.12). Similarly for A.

∎

For Theorem A2.2.14: Since A is nonsingular, I = XA (X is nonsingular). Let P be the row-echelon form of B; then P = FB (F is nonsingular, by Theorem A2.2.4) and P = (FI)B = F(XA)B = (FX)(AB). Since FX is nonsingular* and P is a canonical matrix, P ~ AB and since P ~ B, B & AB have the same rank (Theorem A2.2.6).

∎

For Theorems A2.2.15 & A2.2.16: See Ayres [9], p. 43.

∎

For Theorem A2.2.17: Let r_1, r_2 & r be the ranks of A, B & AB, respectively. Then, $r_1 \le m$, $r_2 \le n < m$ ⟹ $r_2 < m$ and $r \le \mathrm{MIN}\{r_1, r_2\}$ (Theorem A2.2.15), so $r<m$ and hence the m × m matrix AB is singular (Theorem A2.2.12).

∎

For Theorem A2.2.18: Let AB = 0. If any of A or B is nonsingular, appropriate multiplication of AB = 0 by the inverse matrix will leave the other matrix equal to 0; this means that both matrices cannot be nonsingular.

∎

For Theorems A2.2.19, A2.2.20 & A2.2.21: See Blahut [10], pp. 37-9.

∎

* By Theorem A2.2.3 (iii).

APPENDIX 2.4: THE POLYNOMIAL & MATRIX APPROACHES TO CONVOLUTIONAL-CODE THEORY

The 'quantities' in a communications system are the I/P, or the O/P, of its various block units. Each quantity is made of digits denoted by, say, $z_j^{(i)}$, where j denotes time, i denotes input (or output) port and $z_j^{(i)}$ takes values from GF(q) (usually, q=2).

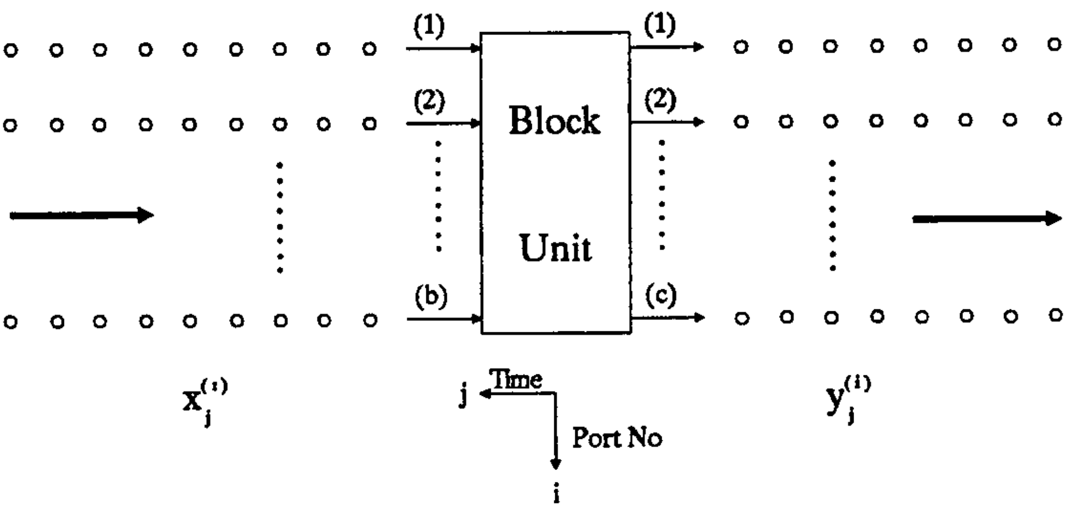


Figure A2.4.1: Organization of digits (o) at the I/P and the O/P of a block unit.

Each block unit has, say, b inputs and c outputs, where b≥1 & c≥1. The input digits $x_j^{(i)}$ and the output digits $y_j^{(i)}$ can be thought of as being organized in a rectangular array. Digits in the same row 'travel' towards (or out of) the same port, while digits in the same column belong to the same time-unit (see Fig. A2.4.1).

In Figs A2.4.1, A2.4.2 & A2.4.3, the little circles (o) represent the digits $x_j^{(i)}$, or $y_j^{(i)}$, and are assumed to flow steadily with time, from left to right. To make mathematical expressions simple, it is necessary to introduce a more compact representation of the $z_j^{(i)}$s; this is achieved by combining the digits either horizontally, or vertically.

In the *matrix approach*, the digits, $z_h^{(1)}$, $z_h^{(2)}$, ..., $z_h^{(d)}$, of column h are combined into a vector $\mathbf{z}_h \triangleq [z_h^{(1)}\ z_h^{(2)} \cdots z_h^{(d)}]$ which represents the input to (or the output of) the block unit at time h (see Fig. A2.4.2a). Subsequent horizontal combination results naturally into a time-sequence of vectors: $z \hat{=} [z_0, z_1, \ldots, z_h, \ldots]$ (see Fig. A2.4.2b). Relations among this type of quantities include infinite-dimensioned matrices, of sub-matrices of approprite dimensions.

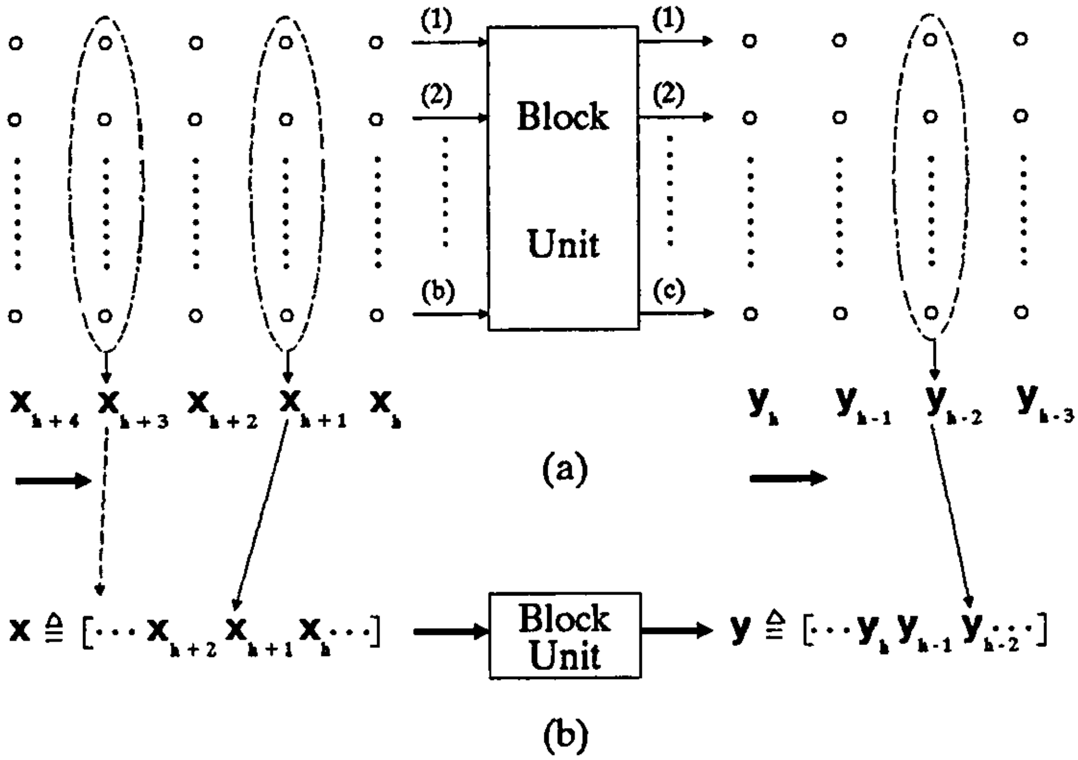


Figure A2.4.2: Matrix approach; formation of a) vectors and b) time sequence of vectors.

In the *polynomial approach*, digits $z_0^{(i)}, z_1^{(i)}, \ldots, z_h^{(i)}, \ldots$ of row i are combined to form a polynomial $z^{(i)}(D) \hat{=} z_0^{(i)} + z_1^{(i)}D + z_2^{(i)}D^2 + \cdots + z_h^{(i)}D^h + \cdots$, which represents the input to (or output of) port i, of the block unit, during all time (see Fig. A2.4.3a). Subsequent vertical combination results naturally, into a vector of polynomials: $Z(D) \hat{=} [z^{(1)}(D), z^{(2)}(D), \ldots, z^{(d)}(D)]$ (see Fig. A2.4.3b). Relations among this type of quantities involve appropriately dimensioned matrices of polynomials.

One advantage of the latter approach is the use of matrices of finite dimensions. The inevitable 'infinite' in convolutional code theory (resulting of course from an infinitely-long message), is contained by the polynomial. Note

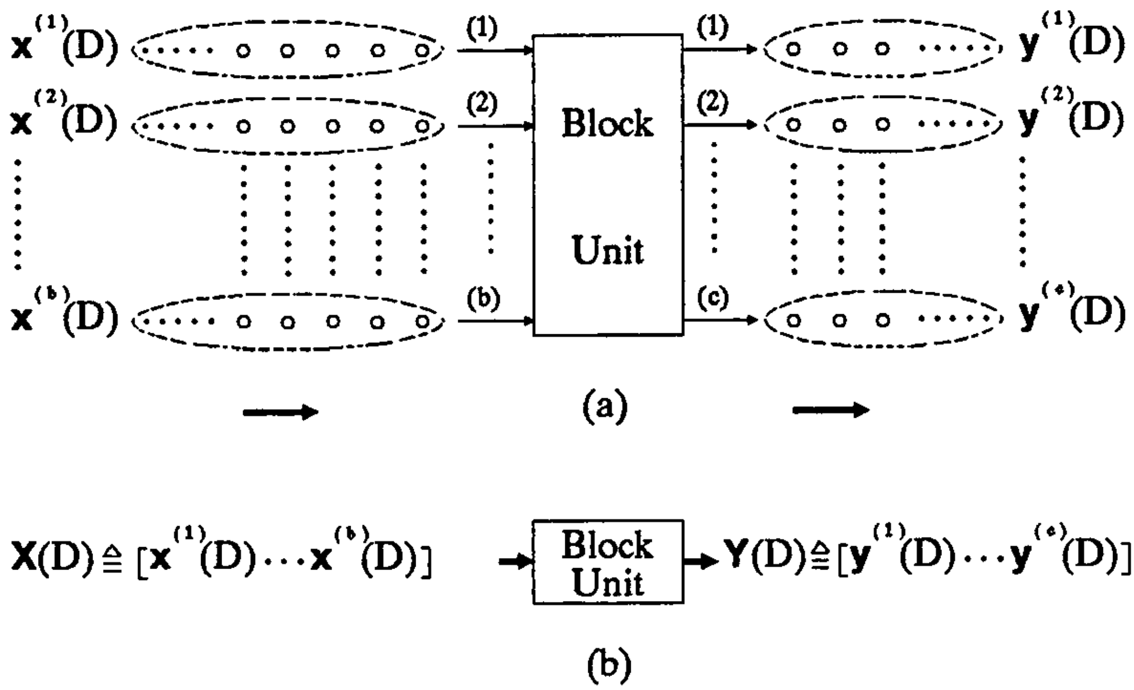


Figure A2.4.3: Polynomial approach; formation of a) polynomials and b) vectors of polynomials.

finally that both z & Z(D), although of different form, represent the same collection of variables $[z_j^{(i)}s]$:

$$z = \Big[\Big(z_0^{(1)}, z_0^{(2)}, \ldots, z_0^{(d)}\Big), \Big(z_1^{(1)}, z_1^{(2)}, \ldots, z_1^{(d)}\Big), \ldots \\ \ldots, \Big(z_h^{(1)}, z_h^{(2)}, \ldots, z_h^{(d)}\Big), \ldots\Big] \qquad (A2.4.1)$$

$$Z(D) = \Big[\ \sum_{h=0}^{+\infty} z_h^{(1)} D^h\ ,\ \sum_{h=0}^{+\infty} z_h^{(2)} D^h\ ,\ \ldots,\ \sum_{h=0}^{+\infty} z_h^{(d)} D^h\ \Big] \qquad (A2.4.2)$$

APPENDIX 2.5: DISTANCE MEASURES FOR CONVOLUTIONAL CODES

Definition A2.5.1: The *ith minimum distance* d_i of a convolutional code is equal to the smallest Hamming distance between any two initial codeword segments, (i+1)-blocks long, that disagree in the initial block [10].

In mathematical language,

if $$\left[u\right]_i \hat{=} \left[u_0,u_1,..,u_i\right] \qquad \text{(A2.5.1a)}$$

and $$\left[v\right]_i \hat{=} \left[v_0,v_1,..,v_i\right] \qquad \text{(A2.5.1b)}$$

then, for i≥0:

$$d_i \hat{=} \text{MIN}\left\{d\Big([v']_i,[v'']_i\Big) : [u']_0 \neq [u'']_0\right\} \qquad \text{(A2.5.2)}$$

∎

The most important distance measure for convolutional codes is the free distance, d_{free}, defined as following:

Definition A2.5.2: If v' & v'' are the codewords corresponding to the information sequences u' & u'', respectively, then the *free distance*, d_{free}, of a convolutional code is defined by

$$d_{free} \hat{=} \text{MIN}\left\{d(v',v'') : u' \neq u''\right\} \qquad \text{(A2.5.3)}$$

If u' and u'' are of unequal length, the shortest is appended with zeros, so that both have equal-length codewords [2].

∎

Another useful distance measure is d_{min}:

Definition A2.5.3: The *minimum distance*, d_{min}, of an (n,k,m) convolutional code is defined to be the mth minimum distance:

$$d_{min} \hat{=} d_m \qquad \text{(A2.5.4)}$$

∎

Much of the earlier work on convolutional codes treated d_{min} as the distance parameter of greatest interest, because the earlier principal decoding techniques had a decoding memory of one constraint-length [2].

Definition A2.5.4: The sequence $d_1,d_2,d_3,...$ is called the *distance profile* of the convolutional code [10].

∎

For convolutional codes that are linear, equations (A2.5.2) & (A2.5.3) can be re-written, using the weight of a binary word:

$$\text{For } i \geq 0: \quad d_i = \text{MIN}\left\{w[v]_i : [u]_0 \neq 0\right\} \tag{A2.5.5}$$

$$d_{free} = \text{MIN}\left\{w(v) : u \neq 0\right\} = \text{MIN}\left\{w(uG) : u \neq 0\right\} \tag{A2.5.6}$$

APPENDIX 2.6: <u>PROOF OF RELATION (2.24)</u>

The following theorem was taken from Noble & Daniel [7]:

<u>Theorem A2.6.1</u>: We can multiply partitioned matrices as if the submatrices were ordinary (scalar) elements, provided that the matrices are partitioned in such a way that the appropriate products can be formed.

∎

Consider relation (2.22):

$$v_h = \left[u_{h-m}, u_{h-m+1}, \ldots, u_h\right] \begin{bmatrix} G_m \\ G_{m-1} \\ \vdots \\ G_0 \end{bmatrix}$$

Note that the message matrix is a 1 x (m+1) one while the system matrix is an (m+1) x 1 one. Consequently, the product of the two will be a single-element matrix (in this case, the elements are submatrices). Note also that the message matrix has been partitioned into (m+1) 1 x k submatrices, while the system matrix has been partitioned into (m+1) k x n submatrices. Hence, the product will be a 1 x n submatrix (as expected):

$$v_h = u_{h-m}G_m + u_{h-m+1}G_{m-1} + \cdots + u_hG_0$$

where $h=0,1,2,\ldots$ and $u_x = 0$ if $x<0$. Then:

$$v_h = \sum_{z=0}^{\theta} u_{h-z}G_z \quad /h=0,1,2,\ldots$$

where $\theta \triangleq \text{MIN}\{m,h\}$.

QED

APPENDIX 2.7: PROOF OF THEOREM 2.3

Consider eqn (2.24):

$$v_h = \sum_{i=0}^{m} u_{h-i} G_i \quad /h=0,1,2,\ldots \quad \& \quad u_j = 0 \text{ if } j<0 \qquad \text{(A)}$$

The objective is to obtain an eqn, similar to the one above, for the channel sequence

$$[v]_z^h \mathrel{\hat{=}} [v_h, v_{h+1}, \ldots, v_{h+z}] \quad /h\geq 0 \;\&\; z\geq 0 \qquad \text{(B)}$$

One way is to increase the limits of the summation, in (A) above, to include all message blocks that participate in the calculation of $[v]_z^h$. h, in (A), can be replaced by h+x, with x ranging from 0 to z:

$$v_{h+x} = \sum_{i=0}^{m} u_{h+x-i} G_i \quad /0\leq x\leq z,\; h\geq 0 \;\&\; u_j = 0 \text{ if } j<0 \qquad \text{(C)}$$

If x is left to range in [0,z] and i in [0,m], then w = h+x-i will range from a maximum of

$$\underset{i}{\mathrm{MAX}}\{\max[h+x-i]\} = \underset{i}{\mathrm{MAX}}\{h+x_{max}-i\} = \underset{i}{\mathrm{MAX}}\{h+z-i\} = h+z-i_{min} = h+z$$

to a minimum of

$$\underset{i}{\mathrm{MIN}}\{\min[h+x-i]\} = \underset{i}{\mathrm{MIN}}\{h+x_{min}-i\} = \underset{i}{\mathrm{MIN}}\{h-i\} = h-i_{max} = h-m$$

So, $w\in[h-m,h+z]$ and substituting in (C), h+x-i = w:

For $x=0,1,\ldots,z$:

$$v_{h+x} = \sum_{w=h-m}^{h+z} u_w G_{h+x-w} \qquad \text{(D)}$$

where: $h\geq 0$, $u_j = 0$ for $j<0$ and $G_j = 0$ for $j\notin[0,m]$.

System (D) can be expanded to:

$$\left.\begin{array}{lllll}
v_h = & u_{h-m}G_m & + u_{h-m+1}G_{m-1} & + \cdots + & u_{h+z}G_{-z} \\
v_{h+1} = & u_{h-m}G_{m+1} & + u_{h-m+1}G_m & + \cdots + & u_{h+z}G_{-z+1} \\
\vdots & \vdots & \vdots & & \vdots \\
v_{h+z} = & u_{h-m}G_{m+z} & + u_{h-m+1}G_{m+z-1} & + \cdots + & u_{h+z}G_0
\end{array}\right\} \longrightarrow \quad \text{(E)}$$

System (E), can be easily written in matrix form to produce eqns (2.25) & (2.26).

APPENDIX 2.8: PROOF OF RELATION (2.36)

Eqn (2.14) gave:

$$v_h^{(j)} = \sum_{w=0}^{m} \sum_{i=1}^{k} u_{h-w}^{(i)} g_{j,w}^{(i)} \quad /h\geq 0,\ 1\leq j\leq n \ \&\ u_x^{(i)} = 0 \text{ for } x<0$$

From the above and eqn (2.35):

$$v^{(j)}(D) = \sum_{h=0}^{+\infty} \left[\sum_{w=0}^{m} \sum_{i=1}^{k} u_{h-w}^{(i)} g_{j,w}^{(i)} \right] D^h \quad /1\leq j\leq n \ \&\ u_x^{(i)} = 0 \text{ if } x<0$$

Interchanging the order of summation:

$$v^{(j)}(D) = \sum_{w=0}^{m} \sum_{h=0}^{+\infty} \sum_{i=1}^{k} u_{h-w}^{(i)} g_{j,w}^{(i)} D^h \quad /1\leq j\leq n \ \&\ u_x^{(i)} = 0 \text{ if } x<0$$

Substituting h=y+w:

$$v^{(j)}(D) = \sum_{w=0}^{m} \sum_{y=-w}^{+\infty} \sum_{i=1}^{k} u_{y}^{(i)} g_{j,w}^{(i)} D^{y+w} \quad /1\leq j\leq n \ \&\ u_x^{(i)} = 0 \text{ if } x<0$$

Because $u_y^{(i)} = 0$, for y<0, y should be non-negative:

$$v^{(j)}(D) = \sum_{w=0}^{m} \sum_{y=0}^{+\infty} \sum_{i=1}^{k} u_{y}^{(i)} g_{j,w}^{(i)} D^{y+w} \quad /1\leq j\leq n$$

Interchanging the order of summation:

$$v^{(j)}(D) = \sum_{i=1}^{k} \left[\sum_{y=0}^{+\infty} u_y^{(i)} D^y \right] \left[\sum_{w=0}^{m} g_{j,w}^{(i)} D^w \right] \quad /1\leq j\leq n$$

By eqn (2.34), the 1st bracket above is $u^{(i)}(D)$, while the 2nd bracket is $g_j^{(i)}(D)$, according to eqn (2.37):

$$v^{(j)}(D) = \sum_{i=1}^{k} u^{(i)}(D) g_j^{(i)}(D) \quad /1\leq j\leq n$$

QED

APPENDIX 2.9: EXAMPLES OF NORMAL-ENCODER CONSTRUCTION

To illustrate the discussion in Section 2.14, three examples are considered. In them, given G(D), the associated normal encoder is constructed.

Example A2.9.1: Consider the generator-polynomial matrix:

$$G(D) = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix}$$

It is obvious that it corresponds to a (3,2,m) code. Since the maximum power of D is 1, then m=1. The normal encoder is made of 2 SRs and 3 X-OR gates. Both SRs have length 1, because the highest power of D along any row of G(D) is 1. The number of non-zero polynomial terms along the three columns (& hence the number of inputs for gates 1,2 &

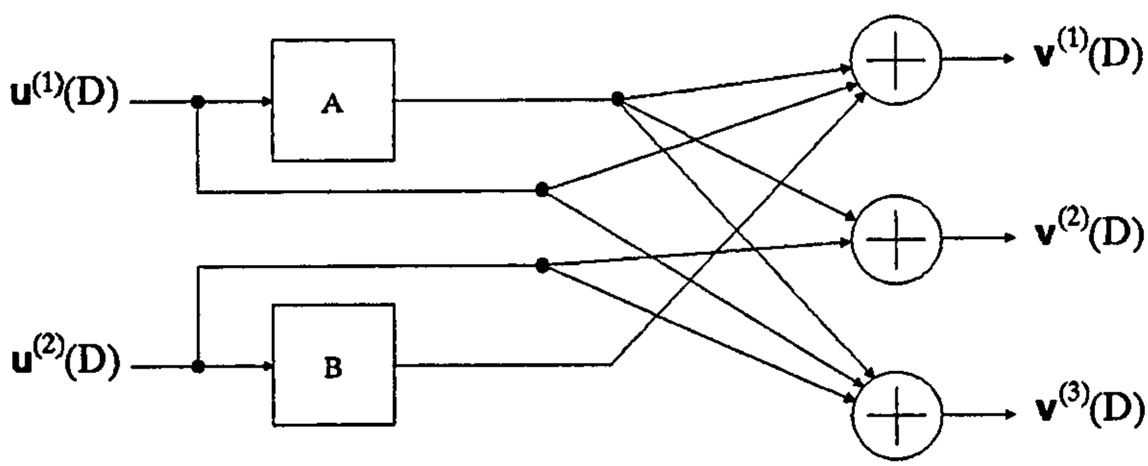


Figure A2.9.1: The normal encoder for a (3,2,1) binary convolutional code.

3) is 3,2 & 3, respectively. The connections are easy to deduce. For example, the contribution to the 3rd gate, from the 1st SR, is the row-1, column-3, polynomial 1+D which indicates two connections, one from the O/P of the 0th stage (i.e. the I/P of the SR) and one from the O/P of the 1st

stage. The diagram of the encoder is shown in Fig. A2.9.1.

∎

Example A2.9.2: Consider the generator-polynomial matrix:

$$G(D) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1+D & D & 1 \\ 0 & D & 1+D^2 & 1+D^2 \end{bmatrix}$$

It is obvious that it corresponds to a (4,3,m) code. Since the maximum power of D is 2, then m=2. The normal encoder is made of 3 SRs and 4 X-OR gates. Note that the highest powers of D along each of the rows of G(D) are 0, 1 & 2; hence these are the lengths of the three SR's. Note also that the number of non-zero polynomial terms along the four columns is 1,4,4, & 4. The connections are easy to deduce. Note finally that an SR of length 0 or a gate with one I/P do not exist. The normal encoder for the above code is shown in Fig. A2.9.2.

∎

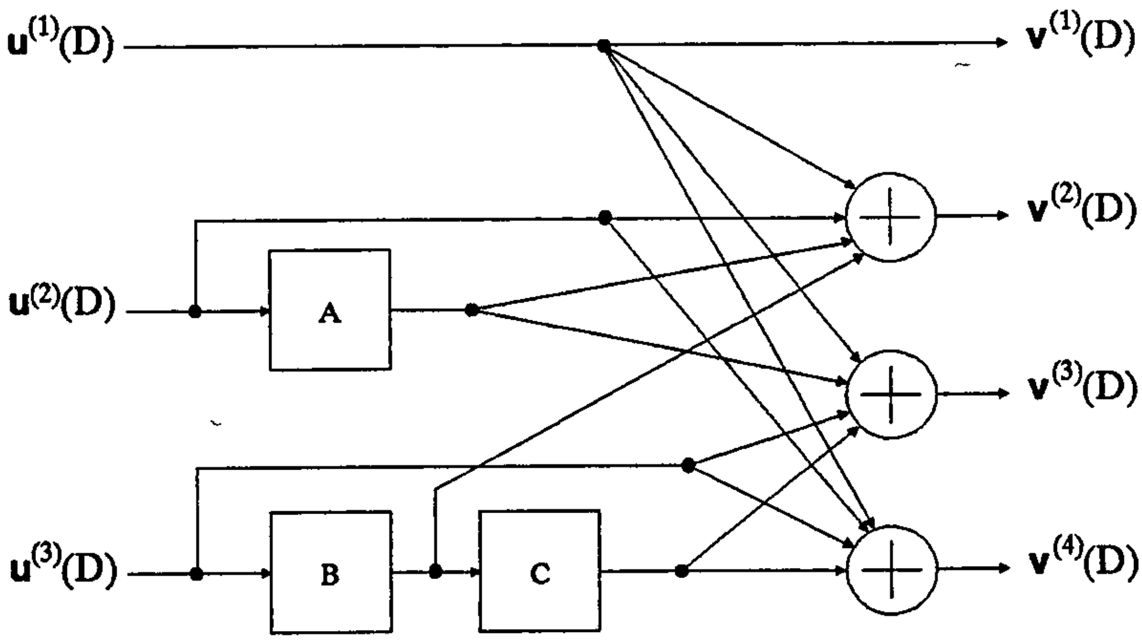


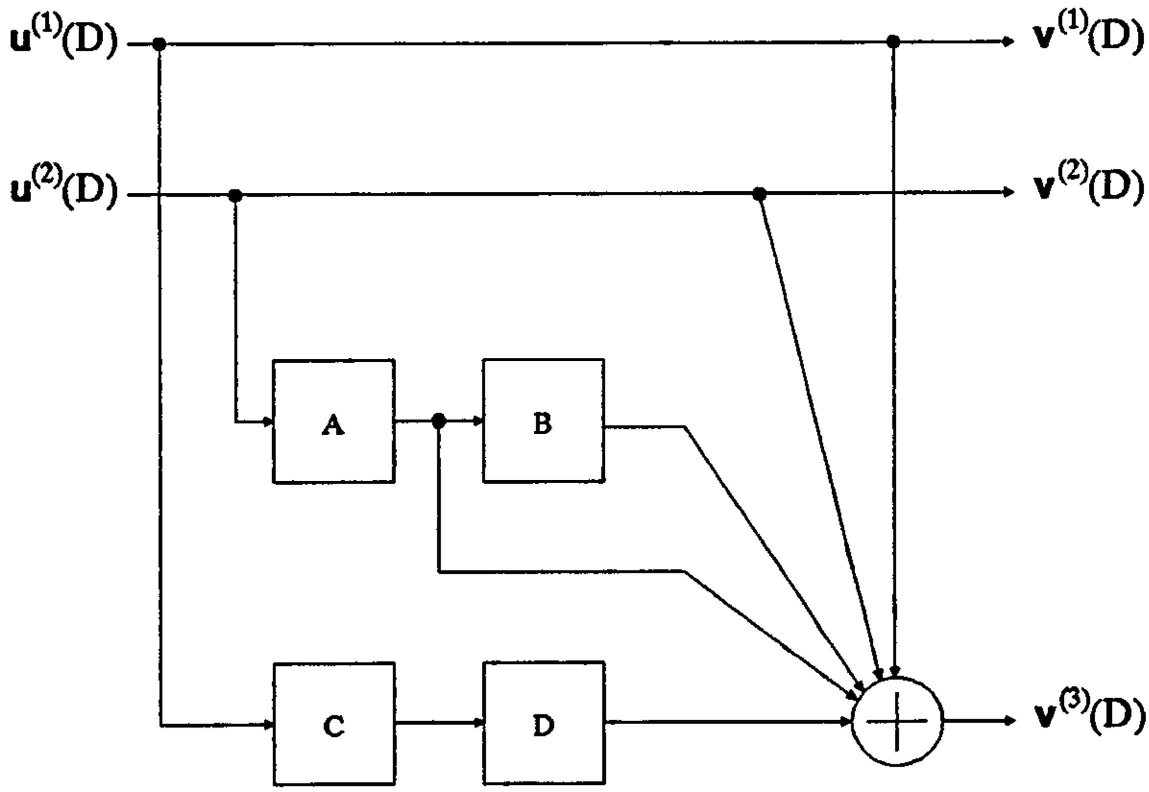
Figure A2.9.2: The normal encoder for a (4,3,2) binary convolutional code.

<u>Example A2.9.3</u>: Consider the (3,2,2) systematic convolutional code with generator-polynomial matrix $G(D) = [I_k, P(D)]$ (for a discussion on the generator-polynomial matrix of systematic codes, see § 2.17.5., p. 39),

where $$P(D) = \begin{bmatrix} 1+D+D^2 \\ 1+D^2 \end{bmatrix}$$

The normal encoder is made of 2 SRs and n-k=1 X-OR gate. Both SRs have length 2 because the highest power of D, along each row of P(D) is 2. The number of non-zero polynomial terms along the only column (and hence the number of I/Ps for the only gate) is 3+2=5. The connections are easy to deduce. The normal encoder for the above code is shown in Fig. A2.9.3.

∎



<u>Figure A2.9.3</u>: The normal encoder for a (3,2,2) binary systematic convolutional code.

APPENDIX 2.10: CATASTROPHIC CODES

Definition A2.10.1: Codes for which an information sequence of infinite Hamming weight may result in a codeword of finite Hamming weight, are called *catastrophic codes* and they are said to suffer from *catastrophic error propagation*. ∎

It has been shown that (see Lin & Costello [2], Sec. 10.3),

For non-catastrophic codes, $\lim_{i \to \infty} \{d_i\} = d_{free}$ (A2.10.1)

Usually, as i increases, d_i reaches d_{free} after 3-4 constraint-lengths [2].

The following theorem, and its proof, appear here in an original form. Nevertheless, the result has been established, long ago, by Massey & Sain [22].

Theorem A2.10.1: A code is non-catastrophic if, and only if, its encoder has a feed-forward (FF) inverse.

Proof:

a) Sufficiency: Let the encoder have an *FF inverse*. This means that there exists an n-input, k-output, linear sequential circuit (LSC), which if it is cascaded with the encoder, they will result in a pure delay-line of h time-units, where h≥0 (Massey & Sain [22], Sec. I).

Assume that the corresponding code is catastrophic. According to Definition A2.10.1 there exists an information sequence **u**, of infinite Hamming weight, which if fed into the encoder, it will generate a codeword, **v**, of finite Hamming weight. If **v** is applied at the inverse, by the nature of the circuit, **u** should be the response. Hence, the inverse is an LSC which produces an infinite sequence (**u**), in response to a finite one (**v**). Hence, this LSC cannot be an FF one. Hence, contradiction.

Then the code is non-catastrophic.

b) Necessity: Let the code be a non-catastrophic one. Since a binary LSC always has an inverse with delay h≥0 (see

Huffman [23], p. 13), the encoder for the above code will also have one.

By eqns (A2.10.1), (A2.5.5) & (A2.5.6), there exists a non-zero k-tuple u_0, such that $u = [u_0,0,0,\ldots,0,\ldots]$ results in a codeword v of weight d_{free}. v is applied at the inverse and after h (h≥0) (block) time-units, it reproduces u_0, while all the subsequent k-tuples are zero. So, this LSC (the inverse) has a transient of only one k-tuple, hence it cannot have feedback loops (LSCs with finite transients have only FF loops - see Huffman [23], p. 7).

QED

The following theorem, due to Massey & Sain ([22], Sec. IV), gives a necessary & sufficient condition for the existence of an FF inverse:

<u>Theorem A2.10.2</u>: A k-input, n-output, feed-forward (FF) linear sequential circuit has an FF inverse either with delay or without delay if, and only if,

$$\gcd\left[\Phi_i(D)\ /i=1,2,\ldots,C(n,k)\right] = D^h \qquad (A2.10.2)$$

for some h≥0, where $\Phi_i(D)$ is the determinant of the ith k x k submatrix of G(D). gcd stands for greatest common divisor, while $C(n,k) \triangleq n!/[k!(n-k)!]$ is the binomial coefficient. Note that there are exactly C(n,k) such submatrices.

∎

Theorem A2.10.2 makes the Massey & Sain [22] paper a classical one, in convolutional code theory. Some authors have defined non-catastrophic codes as those which satisfy eqn (A2.10.2) (see for example Blahut [10], Definition 12.2.3).

The two theorems, given above, imply the existence of an n x k matrix G'(D) such that

$$G(D)G'(D) = I_k D^h \ /h\geq 0 \qquad (A2.10.3)$$

Note also that [by eqns (A2.10.3) & (2.41)]:

$$V(D)G'(D) = U(D)G(D)G'(D) = U(D)D^h \qquad (A2.10.4)$$

Note A2.10.1: Relation (A2.10.4) reveals that any generator-polynomial matrix G(D), satisfying relation (A2.10.2), has an inverse which if multiplied with the channel sequence V(D), it will produce the original message sequence U(D), delayed by h time-units ($h \geq 0$).

∎

Lemma A2.10.1: The generator-polynomial matrix of an (n,k,m) non-catastrophic convolutional code has rank k.

Proof: By eqn (A2.10.3), G(D) has a right inverse:

$$G(D)[(1/D^h)G'(D)] = I_k$$

By Theorem A2.2.11, its rank is k.

QED

Example A2.10.1: Consider the generator-polynomial matrix of Example A2.9.1 (p. 314). The C(3,2) = 3, k x k, submatrices of G(D) mentioned in Theorem A2.10.2, are:

$$\Omega_1(D) = \begin{bmatrix} 1+D & D \\ D & 1 \end{bmatrix} \qquad \Omega_2(D) = \begin{bmatrix} 1+D & 1+D \\ D & 1 \end{bmatrix} \qquad \Omega_3(D) = \begin{bmatrix} D & 1+D \\ 1 & 1 \end{bmatrix}$$

with determinants $\Phi_1(D) = 1+D+D^2$, $\Phi_2(D) = 1+D^2$ and $\Phi_3(D) = 1$. Their greatest common divisor is $(1+D+D^2, 1+D^2, 1) = 1 = D^0$, hence the code of Example A2.9.1 is non-catastrophic and has an FF inverse with no delay.

Consider the output eqns of the circuit of Fig. A2.9.1 (notation is simplified):

$$\left.\begin{aligned} v_1 &= u_1+Du_1+Du_2 \\ v_2 &= Du_1+u_2 \\ v_3 &= u_1+Du_1+u_2 \end{aligned}\right] \longrightarrow (+) \qquad \left.\begin{aligned} v_2+v_3 &= u_1 \\ v_2+Du_1 &= u_2 \end{aligned}\right] \longrightarrow \qquad \begin{aligned} u_1 &= v_2+v_3 \\ u_2 &= v_2+Dv_2+Dv_3 \end{aligned}$$

$$\text{Then:} \quad G'(D) = \begin{bmatrix} 0 & 0 \\ 1 & 1+D \\ 1 & D \end{bmatrix} \quad \text{and} \quad G(D)G'(D) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = I_2D^0$$

Hence, $h = 0$, as predicted.

∎

APPENDIX 2.11: COMPOSITE GENERATOR-POLYNOMIALS

An expression for the encoder's serial output may be obtained by considering that the n (parallel) encoder output-ports are multiplexed for serial transmission (see Fig. 2.1, p. 19). It is obvious that the serial bit-stream must be n times faster than the parallel one.

If X denotes the delay operator for the serial line, then one can write $X^n = D$; this means that successive bits of a particular output must be n time-units apart, in the multiplexed stream. Since also, the zth bit of O/P sequence $v^{(j)}(D)$ is delayed by one time-unit, with respect to the zth bit of $v^{(j-1)}(D)$, sequence $v^{(j)}(D)$ is multiplied by X^{j-1} $(j=1,2,\ldots,n)$. So:

Note A2.11.1: The serial output of the encoder is given by

$$V(X) = v^{(1)}(X^n) + Xv^{(2)}(X^n) + \cdots + X^{n-1}v^{(n)}(X^n) \qquad (A2.11.1)$$

∎

Equation (A2.11.1) can be re-written as

$$V(X) = \sum_{j=1}^{n} X^{j-1}v^{(j)}(X^n)$$

and combined with eqn (2.36):

$$V(X) = \sum_{j=1}^{n} X^{j-1}\left[\sum_{i=1}^{k} u^{(i)}(X^n)g_j^{(i)}(X^n)\right]$$

Interchanging the order of summation:

$$V(X) = \sum_{i=1}^{k} u^{(i)}(X^n)\left[\sum_{j=1}^{n} X^{j-1}g_j^{(i)}(X^n)\right]$$

Finally:

$$V(X) = \sum_{i=1}^{k} u^{(i)}(X^n)g_i(X) \qquad (A2.11.2a)$$

$$\text{where } g_i(X) \triangleq \sum_{j=1}^{n} X^{j-1} g_j^{(i)}(X^n) \quad /i=1,2,\ldots,k \qquad \text{(A2.11.2b)}$$

<u>Definition A2.11.1</u>: The k polynomials $g_i(X)$ /i=1, 2,...,k, defined by eqn (A2.11.2b), are called *composite generator-polynomials* [2].

∎

<u>Note A2.11.2</u>: The ith ($1 \le i \le k$) composite generator-polynomial relates the ith input sequence to the serial encoder output.

∎

<u>Example A2.11.1</u>: Consider now the encoder of Example A2.9.1 (p. 314). This is a (3,2,1) code, hence it has k = 2 composite generator-polynomials, which may be obtained from (A2.11.2b).

From Example A2.9.1 and the form of G(D) [see reln (2.41d), p. 33)], the following eqn is obtained:

$$G(D) = \begin{bmatrix} g_1^{(1)}(D) & g_2^{(1)}(D) & g_3^{(1)}(D) \\ g_1^{(2)}(D) & g_2^{(2)}(D) & g_3^{(2)}(D) \end{bmatrix} = \begin{bmatrix} 1+D & D & 1+D \\ D & 1 & 1 \end{bmatrix} \qquad \text{(A)}$$

From (A), substituting $D = X^3$ (n = 3):

$$\left.\begin{array}{lll} g_1^{(1)}(X^3) = 1+X^3 & g_2^{(1)}(X^3) = X^3 & g_3^{(1)}(X^3) = 1+X^3 \\ g_1^{(2)}(X^3) = X^3 & g_2^{(2)}(X^3) = 1 & g_3^{(2)}(X^3) = 1 \end{array}\right] \longrightarrow \qquad \text{(B)}$$

Using (B) in reln (A2.11.2b):

$$\left.\begin{array}{l} g_1(X) = X^0(1+X^3) + X^1(X^3) + X^2(1+X^3) = 1+X^2+X^3+X^4+X^5 \\ g_2(X) = X^0(X^3) + X^1(1) + X^2(1) = X+X^2+X^3 \end{array}\right] \longrightarrow \text{(C)}$$

Using (C) in eqn (A2.11.2a), the encoder's serial output is obtained in terms of its two inputs:

$$V(X) = u^{(1)}(X^3)(1+X^2+X^3+X^4+X^5) + u^{(2)}(X^3)(X+X^2+X^3) \qquad \text{(D)}$$

To verify the correctness of (D), consider the following simple input:

$$U(X) = 1+X^2+X^5 \qquad (E)$$

(E) corresponds to the message sequence

$$u = (1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ \cdots) \qquad (F)$$

from which, the two inputs are obtained by demultiplexing into two streams:

$$u^{(1)}(D) = 1+D \quad \text{and} \quad u^{(2)}(D) = D^2 \qquad (G)$$

Substituting $D = X^3$ in (G) and then into (D), the serial response of the encoder [i.e. the three multiplexed output bit streams, in response to the input (G)] is:

$$V(X) = (1+X^3)(1+X^2+X^3+X^4+X^5) + (X^6)(X+X^2+X^3) \longrightarrow$$

$$V(X) = 1+X^2+X^4+X^6+X^9 \qquad (H)$$

(H) corresponds to the channel sequence

$$v = (1\ 0\ 1\ 0\ 1\ 0\ 1\ 0\ 0\ 1\ 0\ 0\ \cdots) \qquad (I)$$

From Example A2.10.1, the eqns for the encoder O/P are:

$$\left.\begin{aligned} v^{(1)}(D) &= (1+D)u^{(1)}(D) + Du^{(2)}(D) \\ v^{(2)}(D) &= Du^{(1)}(D) + u^{(2)}(D) \\ v^{(3)}(D) &= (1+D)u^{(1)}(D) + u^{(2)}(D) \end{aligned}\right\} \longrightarrow \qquad (J)$$

Substituting the input (G) into eqns (J):

$$v^{(1)}(D) = (1+D)(1+D)+DD^2 = 1+D^2+D^3 \longleftrightarrow 1\ 0\ 1\ 1\ 0\ \cdots$$
$$v^{(2)}(D) = D(1+D) + D^2 = D \longleftrightarrow 0\ 1\ 0\ 0\ 0\ \cdots$$
$$v^{(3)}(D) = (1+D)(1+D)+D^2 = 1 \longleftrightarrow 1\ 0\ 0\ 0\ 0\ \cdots$$

multiplex

$\longrightarrow$ 1 0 1 0 1 0 1 0 0 1 0 0 0 0 0 ···

Comparing the last result, with (I), it is indeed verified that V(X) gives the serial channel bit-stream of the (parallel-in, parallel-out) encoder.

∎

APPENDIX 2.12: PROOF OF THEOREM 2.5

Using Definition 2.10, one may write:

$$v_h = \left[u_h, v_h^{(p)}\right] \quad /h>0 \quad \& \quad v_h^{(p)} \text{ is an } (n-k)\text{-tuple} \qquad (A)$$

Consider also the partition of the k x n matrix G_z [defined by (2.21), p. 27]:

$$G_z = \left[G_z', P_z\right] \qquad (B)$$

where G_z' is k x k and P_z is k x (n-k).

Using eqns (A) & (B), in eqn (2.24), with $\theta \triangleq \text{MIN}\{h,m\}$:

$$\left[u_h, v_h^{(p)}\right] = \sum_{z=0}^{\theta} u_{h-z}\left[G_z', P_z\right] \qquad (C)$$

Using Theorem A2.6.1 (p. 311), about the multiplication of partitioned matrices, on eqn (C):

$$u_h = \sum_{z=0}^{\theta} u_{h-z} G_z' \quad \Longleftrightarrow \quad u_h = u_h G_0' + \sum_{z=1}^{\theta} u_{h-z} G_z'$$

$$\Longleftrightarrow \quad u_h\left(I_k + G_0'\right) + \sum_{z=1}^{\theta} u_{h-z} G_z' = 0$$

For the above eqn to hold true for all messages u_i, all the coefficients of u_i must be zero:

$$\left.\begin{array}{l} I_k + G_0' = 0 \quad \Longleftrightarrow \quad G_0' = I_k \\ \qquad\qquad\qquad\qquad\quad G_z' = 0 \quad /z\geq 0 \end{array}\right\} \longrightarrow \qquad (D)$$

Theorem 2.5 follows from eqns (B) & (D).

QED

APPENDIX 2.13: COMPOSITE GENERATOR-POLYNOMIALS FOR SYSTEMATIC CONVOLUTIONAL CODES

Applying the results of Lemma 2.8, to the definition of composite generator-polyomials, one gets:

Theorem A2.13.1: For an (n,k,m) systematic convolutional code, the composite generator-polynomials have the following form:

$$g_i(X) = X^{k-1}\Big[\sum_{j=1}^{n-k} X^j g_{k+j}^{(i)}(X^n)\Big] + X^{i-1} \quad /i=1,2,\ldots,k \qquad (A2.13.1)$$

Proof: From eqn (A2.11.2b), for i=1,2,...,k:

$$g_i(X) \triangleq \sum_{z=1}^{n} X^{z-1} g_z^{(i)}(X^n)$$

$$\Longrightarrow \qquad g_i(X) = \sum_{z=1}^{k} X^{z-1} g_z^{(i)}(X^n) + \sum_{z=k+1}^{n} X^{z-1} g_z^{(i)}(X^n)$$

From relation (2.49) (p. 38), $g_1^{(i)}(D)$, $g_2^{(i)}(D)$, ..., $g_{i-1}^{(i)}(D)$, $g_{i+1}^{(i)}(D)$, ..., $g_k^{(i)}(D)$ are all zero and $g_i^{(i)}(D) = 1$ for all i=1,2, ...,k. Using this, in the 1st summation of the above eqn and substituting z = k+j in the 2nd summation, eqn (A2.13.1) is obtained.

QED

APPENDIX 2.14: EXAMPLE OF A TYPE-II ENCODER

Example A2.14.1: Consider the systematic code of Example A2.9.3 (p. 316). Its generator-polynomial submatrix P(D), is enough to generate the type-II encoder:

Since n-k=1 there is only one SR. Since the maximum exponent of D in P(D) is 2, the SR has length 2 (M_1=2).

Since the number of 'ones', along the column of P(D) is two, then the 0th gate has a total of 3 I/Ps.

Since the number of Ds, along the column of P(D) is one, then the 1st gate has a total of 2 I/Ps.

Since the number of D^2s, along the column of P(D) is two, then the 2nd (& last) gate has a total of 2 I/Ps.

Connections are easy to determine. For example, looking along the 1st row, one sees three terms; this means that $u^{(1)}(D)$ contributes to all three gates. Along the 2nd row [for connections from $u^{(2)}(D)$] there are the terms 1 & D^2. "1" means a connection to the 0th gate, while "D^2" means a connection to the 2nd gate. The encoder is shown in Fig. A2.14.1.

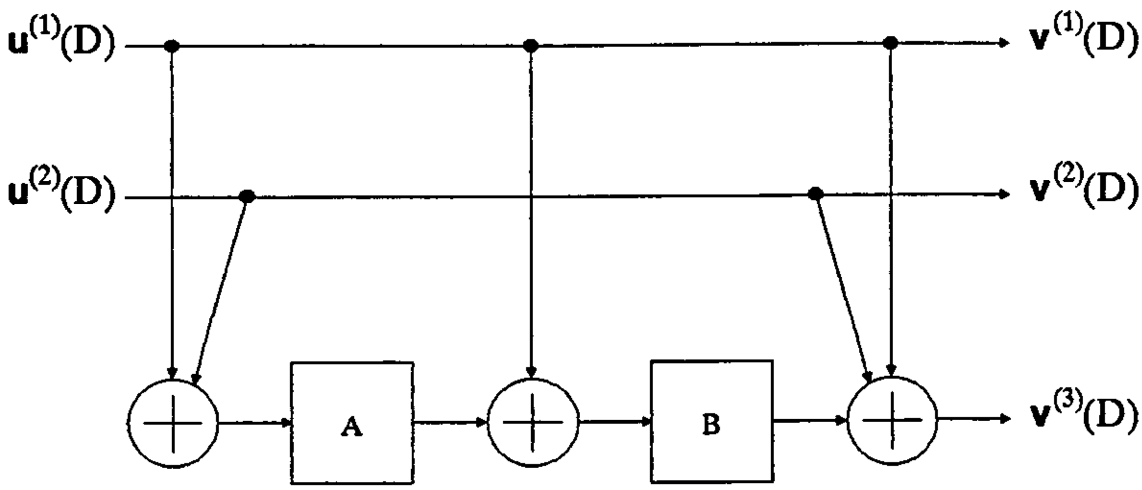


Figure A2.14.1: Type-II encoder for the (3,2,2) systematic code of Example A2.9.3. The normal encoder, for the same code, is illustrated in Fig. A2.9.3.

APPENDIX 2.15: PROOF OF THEOREM 2.10

Condition (2.54) and the partition of G(D), instruct the following partition for $H^T(D)$ (see also Theorem A2.6.1):

$$H^T(D) = \begin{bmatrix} Y(D) \\ Z(D) \end{bmatrix} \qquad (A)$$

where $Y(D)$ is a $k \times (n-k)$ matrix
and $Z(D)$ is an $(n-k) \times (n-k)$ matrix.

Combining eqn (2.54) with eqn (A):

$$\left[I_k, P(D)\right] \begin{bmatrix} Y(D) \\ Z(D) \end{bmatrix} = I_k Y(D) + P(D)Z(D) = 0 \implies$$

$$\implies \quad Y(D) = -P(D)Z(D) \quad \text{and substituting in (A):}$$

$$H^T(D) = \begin{bmatrix} -P(D)Z(D) \\ Z(D) \end{bmatrix} \qquad \text{and using Theorem A2.2.1:}$$

$$H(D) = \left[-Z^T(D)P^T(D), Z^T(D)\right] = Z^T(D)\left[-P^T(D), I_{n-k}\right] \qquad (B)$$

The rank of H(D) (n-k) cannot exceed the rank of $Z^T(D)$, or $[-P(D), I_{n-k}]$ (see Theorem A2.2.15, p. 303) and since both have n-k rows they should both have rank n-k. Since $Z^T(D)$ is a square matrix, it must be non-singular (see Theorem A2.2.12, p. 303)

QED

APPENDIX 2.16: PROOF OF THEOREM 2.12

A constructive proof of Theorem 2.11 can be obtained, if H is seen as the limit of $[H]_z$ /z—>+∞. Let $h_{i,j} = [X_{i,j}, Y_{i,j}]$ ($0 \le i \le z$ & $0 \le j \le z$) where, $X_{i,j}$ is an (n-k) x k matrix and $Y_{i,j}$ is an (n-k) x (n-k) matrix. Then, condition (2.58) gives [see (2.43) & (2.25c)]:

for z=0: $$X^T_{0,0} + P_0 Y^T_{0,0} = 0 \qquad (A)$$

while for z>0:

$$[G]^0_z [H]^T_z = \begin{bmatrix} [G]^0_{z-1} & K_z \\ 0 & G_0 \end{bmatrix} \begin{bmatrix} [H]^T_{z-1} & R^T_z \\ C^T_z & h^T_{z,z} \end{bmatrix} = 0 \qquad (B)$$

where 0 is a 1 x z matrix of k x n submatrices,

$$K^T_z \triangleq \left[G^T_z, G^T_{z-1}, \ldots, G^T_1\right]$$

$$R_z \triangleq \left[h_{z,0}, h_{z,1}, \ldots, h_{z,z-1}\right]$$

and $$C^T_z \triangleq \left[h^T_{0,z}, h^T_{1,z}, \ldots, h^T_{z-1,z}\right]$$

with $G_z = 0$, if z>m.

From eqn (B), using (A) & (2.58):

$$K_z C_z^T = 0 \qquad \text{(Ca)}$$

$$G_0 C_z^T = 0 \qquad \text{(Cb)}$$

$$G_0 h_{z,z}^T = 0 \qquad \text{(Cc)}$$

$$\left[G\right]_{z-1}^{0} R_z^T + K_z h_{z,z}^T = 0 \qquad \text{(Cd)}$$

The system of equations (C) will serve as the set of conditions, $[H]_z$ has to satisfy. If the equals of K_z, C_z^T and R_z^T are used in system (C), together with the matrix partitions $G_0 = [I_k, P_0]$, $G_i = [0_k, P_i]$ $(i=1,2,\dots,m)$ and $h_{i,j} = [X_{i,j}, Y_{i,j}]$, the following results are obtained:

$$P_i^T Y_{j,z}^T = 0 \quad /i=1,2,\dots,z \;\; \& \;\; j=0,1,\dots,z-1 \qquad \text{(Da)}$$

$$X_{j,z}^T + P_0 Y_{j,z}^T = 0 \quad /j=0,1,\dots,z-1 \qquad \text{(Db)}$$

$$X_{z,z}^T + P_0 Y_{z,z}^T = 0 \qquad \text{(Dc)}$$

$$\left[G\right]_{z-1}^{0} R_z^T = -K_z h_{z,z}^T \qquad \text{(Dd)}$$

A solution for eqn (a) is $Y_{j,z}^T = 0$ and this combined with eqn (b), gives $X_{j,z}^T = 0$, so that $h_{j,z} = 0$ $/j=0,1,\dots,z-1$, and hence $C_z = 0$.

Eqn (c) will determine $h_{z,z}$; one solution is $h_{z,z} = Y_{z,z}[-P_0^T, I_{n-k}]$, where $Y_{z,z}$ is any nonsingular $(n-k) \times (n-k)$ matrix; usually, $Y_{z,z} = I_{n-k}$.

Eqn (d) will determine R_z^T; it can be rewritten as:

$$X_{z,j}^T + P_0 Y_{z,j}^T + \sum_{i=1}^{z-1-j} P_i Y_{z,i+j}^T = -P_{z-j} \quad /j=0,1,\dots,z-1 \qquad \text{(E)}$$

One solution for (E) is $Y_{z,j} = 0$ $/j=0,1,\dots,z-1$. This gives $X_{z,j}^T = -P_{z-j}$ $/j=0,1,\dots,z-1$, so that:

$$h_{z,j} = \left[-P_{z-j}^T, 0\right] \quad /j=0,1,\dots,z-1$$

Finally:

$$R_z = -\left[P_z^T, 0, P_{z-1}^T, 0, \dots, P_1^T, 0\right] \qquad \text{(F)}$$

The above result concludes the construction. Note that the $[H]_z$ obtained, is not unique.

APPENDIX 2.17: PROOF OF THEOREM 2.15

Substitute $s^{(j)}(D)$ [from eqn (2.72)], $e^{(i)}(D)$ [from eqn (2.69)] and $g_{k+j}^{(i)}(D)$ [from eqn (2.37)], in eqn (2.75):

For j=1,2,...,n-k:

$$\sum_{h=0}^{+\infty} s_h^{(j)} D^h = -\sum_{i=1}^{k} \sum_{y=0}^{+\infty} \sum_{z=0}^{m} e_y^{(i)} g_{k+j,z}^{(i)} D^{y+z} + \sum_{h=0}^{+\infty} e_h^{(k+j)} D^h \longrightarrow$$

$$\sum_{h=0}^{+\infty} s_h^{(j)} D^h = -\sum_{i=1}^{k} \sum_{h=0}^{+\infty} \sum_{z=0}^{m} e_{h-z}^{(i)} g_{k+j,z}^{(i)} D^h + \sum_{h=0}^{+\infty} e_h^{(k+j)} D^h \longrightarrow$$

$$\sum_{h=0}^{+\infty} s_h^{(j)} D^h = \sum_{h=0}^{+\infty} \left\{ e_h^{(k+j)} - \sum_{z=0}^{m} \sum_{i=1}^{k} e_{h-z}^{(i)} g_{k+j,z}^{(i)} \right\} D^h \longrightarrow$$

$$s_h^{(j)} = e_h^{(k+j)} - \sum_{z=0}^{m} \sum_{i=1}^{k} e_{h-z}^{(i)} g_{k+j,z}^{(i)} \quad /h=0,1,2,\ldots$$

where $e_x^{(i)}=0$ if $x<0$, or otherwise:

$$s_h^{(j)} = e_h^{(k+j)} - \sum_{z=0}^{\theta} \sum_{i=1}^{k} e_{h-z}^{(i)} g_{k+j,z}^{(i)} \quad /h=0,1,2,\ldots$$

where $\theta \triangleq \mathrm{MIN}\{h,m\}$.

The expression in terms of $r_h^{(i)}$ is obtained in exactly the same way.

QED

APPENDIX 3.1: SEQUENTIAL MACHINES & STATE TRANSITIONS

The following definitions are taken from Booth [6] (chapter 3):

Definition A3.1.1: A *sequential machine* is a system that has the following properties:

i) Its internal behavior is described in terms of a set, Q, of possible states the system might enter.
ii) The possible inputs to the system are assumed to be sequences of symbols selected from a finite set, I, of input symbols.
iii) The possible outputs of the system are assumed to be sequences of symbols selected from a finite set, Z, of output symbols.
iv) The system produces an output symbol whenever an input symbol is applied.

∎

Definition A3.1.2: A sequential machine is called a *Mealey machine* if it is characterized by the following:

i) A set of Q states.
ii) A finite set, I, of input symbols.
iii) A finite set, Z, of output symbols.
iv) A mapping*, f, of $I \times Q$ into Q, called the *next-state function*.
v) A mapping, g, of $I \times Q$ onto Z, called the *output function*.

A particular machine is denoted by the 5-tuple $\langle I, Q, Z, f, g \rangle$.

∎

Definition A3.1.3: A sequential machine is called a *Moore machine* if it differs from a Mealey machine only in that its output mapping g is restricted to a mapping of Q onto Z.

∎

* For a definition of the various types of mapping, see Definition A2.1.14.

<u>Note A3.1.1</u>: *Transition diagrams* provide a graphical representation of the operation of a machine. Each diagram consists of a set of labelled boxes (or circles) that correspond to the states of the machine.

For each ordered pair of states S_a and S_b a *directed edge* will connect state S_a to state S_b if, and only if, there exists an input symbol i_a in I such that $f(i_a,S_a) = S_b$.

If a directed edge connects state S_a to state S_b when the input is i_a, the *edge is labelled* as $i_a/g(i_a,S_a)$.

The boxes (or circles) of the transition diagram correspond to the current state of the system; the label on the edge indicates the current input and the current output. The arrowhead on each edge indicates the next state of the machine.

If more than one input symbols cause a specific transition from, say, S_a to S_b then a *multiple-edge* representation is used: A single directed line with a multiple label.

∎

<u>Example A3.1.1</u>: Consider the (3,2,1) encoder of Fig. A2.9.1 (p. 314). Its state diagram contains 4 states, S_0, S_1, S_2 & S_3. Due to the special configuration of this encoder, it is easy to construct the state diagram. Note that the encoder memory is completely reset after each transition, because the 'depth' of its SRs is only one. This means that the current I/P block u_h will become the next state. From the encoder circuit-diagram, the following equations are obtained (in simplified notation):

$v_1 = u_1 + A + B$ $\qquad v_2 = u_2 + A$ $\qquad v_3 = u_1 + u_2 + A$

Current state $= S = [BA]$ $\qquad$ *Next state* $= S' = [u_2u_1]$

For each of the four current states S, the above eqns are modified for the particular values of A & B (as shown below). Following that, each of the four sets of simplified eqns is used to produce the next state and the output, by letting $[u_1u_2]$ assume each of its four values.

	$S = [BA]$		$S = [BA]$		$S = [BA]$		$S = [BA]$	
	$S_0 = [00]$		$S_1 = [01]$		$S_2 = [10]$		$S_3 = [11]$	
	$v_1 = u_1$		$v_1 = \bar{u}_1$		$v_1 = \bar{u}_1$		$v_1 = u_1$	
	$v_2 = u_2$		$v_2 = \bar{u}_2$		$v_2 = u_2$		$v_2 = \bar{u}_2$	
	$v_3 = u_1 + u_2$		$v_3 = \overline{u_1 + u_2}$		$v_3 = u_1 + u_2$		$v_3 = \overline{u_1 + u_2}$	
$u_1 u_2$	S'	$v_1 v_2 v_3$	S'	$v_1 v_2 v_3$	S'	$v_1 v_2 v_3$	S'	$v_1 v_2 v_3$
0 0	S_0	0 0 0	S_0	1 1 1	S_0	1 0 0	S_0	0 1 1
0 1	S_2	0 1 1	S_2	1 0 0	S_2	1 1 1	S_2	0 0 0
1 0	S_1	1 0 1	S_1	0 1 0	S_1	0 0 1	S_1	1 1 0
1 1	S_3	1 1 0	S_3	0 0 1	S_3	0 1 0	S_3	1 0 1

The results are summarized in the encoder state-transition diagram of Fig. A3.1.1:

∎

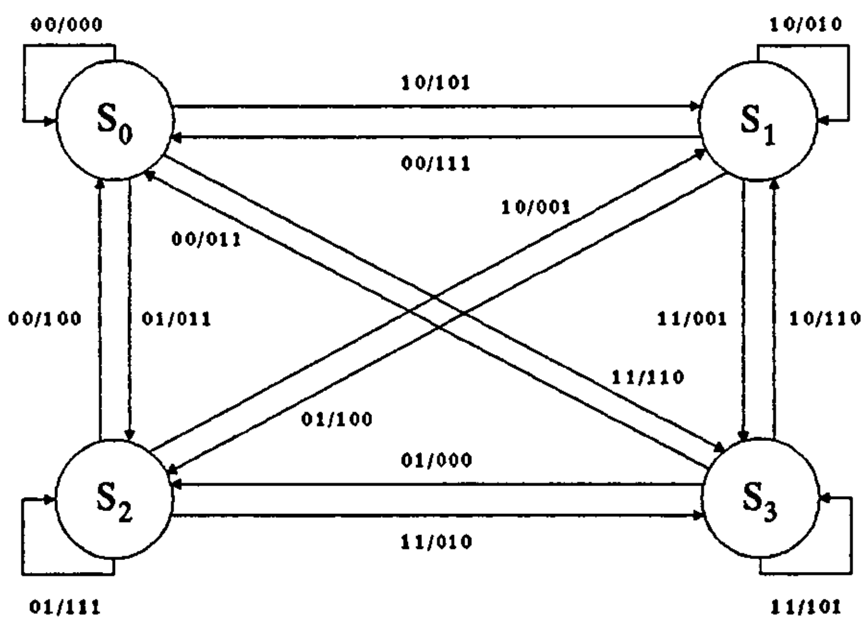


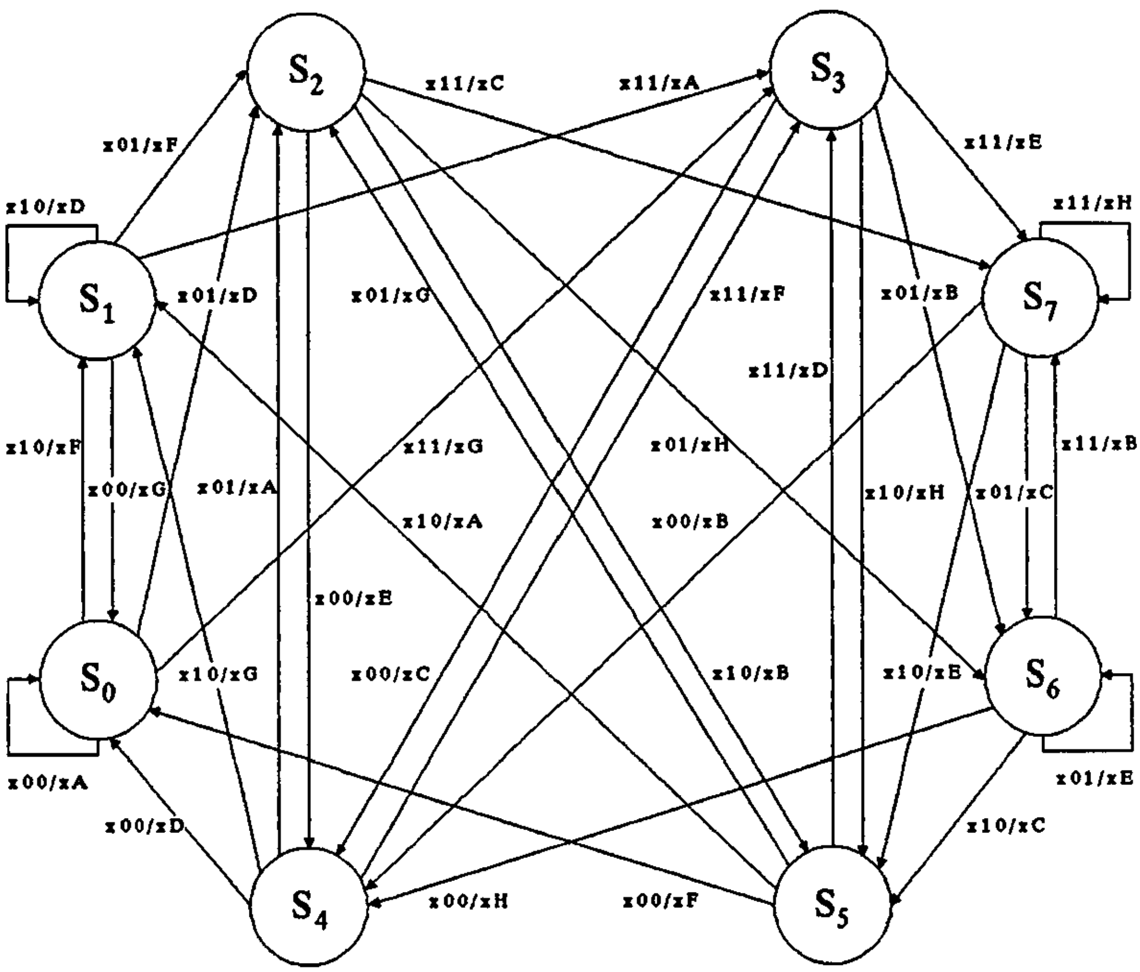
Figure A3.1.1: State-transition diagram for the (3,2,1) normal encoder of Fig. A2.9.1 (p. 314).

Example A3.1.2: Consider the normal (4,3,2) encoder of Fig. A2.9.2 (p. 315). It has 3 SR stages, hence 8 states (see Example 3.2, p. 56). The next-state and output eqns are obtained from the circuit diagram, and are shown below:

$v_1 = u_1$ $\qquad$ $v_2 = u_1 + u_2 + A + B$

$v_3 = u_1 + u_3 + A + C$ $\qquad\qquad$ $v_4 = u_1 + u_2 + u_3 + C$

Current state = S = [CBA] $\qquad\qquad$ *Next state* = S' = $[Bu_3u_2]$



x = 0, or 1

A —> xxx $\quad$ B —> $xx\bar{x}$ $\quad$ C —> $x\bar{x}x$ $\quad$ D —> $x\bar{x}\bar{x}$

E —> $\bar{x}xx$ $\quad$ F —> $\bar{x}x\bar{x}$ $\quad$ G —> $\bar{x}\bar{x}x$ $\quad$ H —> $\bar{x}\bar{x}\bar{x}$

Figure A3.1.2: State-transition diagram for the (4,3,2) normal encoder of Fig. A2.9.2. All labels are double-edge ones (one for each value of x). $u_h^{(1)} = x = 0$ or 1. The output is a logical function of x and has the form xXYZ, where XYZ $\in$ {A,B,...,H}.

For each of the eight current states S = [CBA], the set of eqns above is simplified, and then used to obtain the next state and the output, by letting $[u_1u_2u_3]$ assume its eight possible values. The resulting state-transition diagram is shown in Fig. A3.1.2.

∎

Example A3.1.3: Consider finally, the type-II encoder of Fig. A2.14.1 (p. 325). Its next-state and output eqns are:

$$v_1 = u_1 \quad v_2 = u_2 \quad v_3 = u_1 + u_2 + B$$

Current state = S = [BA] *Next state* = S' = [$(u_1$+A)(u_1+u_2)]

Following the same procedure as before, the state-transition diagram of Fig. A3.1.3 is produced.

∎

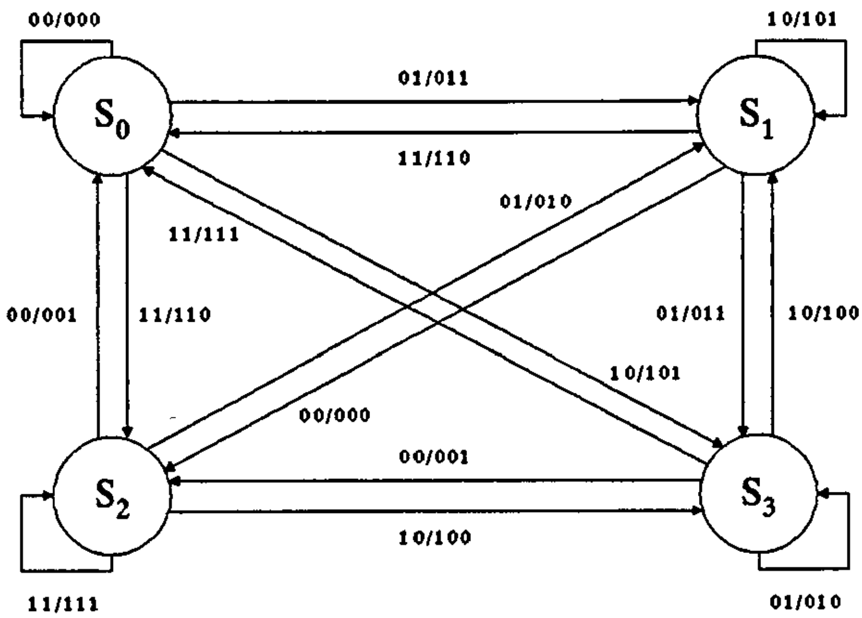


Figure A3.1.3: State-transition diagram for the (3,2,2) type-II encoder of Fig. A2.14.1 (p. 325).

APPENDIX 3.2: PROOF OF THEOREMS 3.1, 3.2 & 3.3

A3.2.1. Proof of Theorem 3.1

According to Note 2.9, the ith row of the generator-polynomial matrix G(D), determines the contributions (to the encoder O/P) from the ith SR ($1 \le i \le k$). In particular, a connection from the hth stage ($0 \le h \le M_i$) of the ith SR to the jth X-OR gate ($1 \le j \le n$) exists, iff the coefficient of D^h in $g_h^{(i)}(D)$ is non-zero. It follows easily then that the existence of a non-zero coefficient for D^h, in any of the polynomials of the ith row, implies that the O/P of the hth stage of the ith SR contributes to the encoder O/P. By the same token, the highest power in the polynomials of the ith row, is M_i; consequently, if this highest-power term is 1 then $M_i = 0$. So, for the normal encoder, $f[G(D)]$ equals the number of zero-length SRs.

QED

A3.2.2. Proof of Theorem 3.2

According to Theorem 3.1, f of the k SRs* have zero length (i.e. are non-existent), say SRs number $\alpha(1)$, $\alpha(2),\ldots,\alpha(f)$. Then f of the k input digits, specifically digits $u_h^{[\alpha(1)]},u_h^{[\alpha(2)]},\ldots,u_h^{[\alpha(f)]}$ cannot be stored in the memory of the encoder, hence they do not participate in the formation of the new encoder state. As a consequence, there are q^{k-f} different ways of altering the encoder state in a single time-unit, so in a state-transition diagram there are q^{k-f} transitions out of each state.

Let the next state be $S(h+1)=S_n$. S_n can be reached from the current state S(h) within a single time-unit. How many states can 'act' as current state S(h)? Or, to put it otherwise, what are the restrictions on S(h) so that the next state is S_n?

Note from Fig. 3.2 that to reach S_n with one transition, F(h) & C(h) (the current state of FEG & CEG, respectively) must have a unique and specific composition, because they will form C(h+1) & R(h+1) - the next state of the CEG & REG, respectively; to be precise, if FEG$\cap$REG $\neq \emptyset$ then the digits of F(h) that are common with the ones of R(h) can assume any value. In contrast, R(h) may have any composition (during the transition this group will leave the encoder).

So the format of the current state S(h), from which S_n can be reached with one transition, is: "Specific F(h) & C(h) and any R(h)". Since R(h) contains k-f digits (see Definition 3.1), there are q^{k-f} states from which another state can be reached with one transition.

Consider now the labelling of each transition with the

* A q-ary encoder is made of q-ary SR stages and GF(q) gates.

I/P block (a k-tuple) that caused it. It was mentioned earlier that f of the k source digits cannot be stored in the encoder memory and hence they do not participate in the formation of the next state. This means that for each $(k-f)$-tuple that causes a state transition there are f I/P digits that can have any value, hence to any transition there correspond q^f source blocks.

QED

A3.2.3. Proof of Theorem 3.3

According to Theorems 3.2 & 3.1, there are q^{k-f} transitions entering any particular state*. Consider the transition $S(h) \longrightarrow S(h+1)$. What are the restrictions on u_h, the I/P k-tuple at time-unit h, if the next state is S_n?

It is obvious from Fig. 3.2 (p. 59) that $F(h+1)=F_n$ depends entirely on u_h. Specifically, all the $k-f$ digits of u_h that correspond to SRs of length more than one (i.e. those that will reside in the circuit memory during the next time-unit) will form F_n. Consequently, these $k-f$ digits are completely specified, once S_n is given. By the same token, though, the rest f digits can have any value.

The conclusion from the above discussion is that in order for the next state, $S(h+1)$, to be state S_n, only f of the k digits of the current I/P block u_h can be chosen freely (the rest are determined by F_n); hence there are q^f different I/P blocks that can trigger the previously considered transition. The above conclusion holds true for the transition $S(h) \longrightarrow S_n$, which means that it holds true for all the q^{k-f} states that can change to S_n.

QED

APPENDIX 3.3: EXAMPLE OF TRELLIS DIAGRAM

Example A3.3.1: Consider the code with generator-polynomial matrix $G(D) = [1+D^2 \;\; 1+D+D^2]$. Its transition diagram is shown in Fig. A3.3.1. The trellis diagram follows readily (Fig. A3.3.2). Note that the central portion of the trellis extends from time-unit 2 to time-unit 7.

* A q-ary encoder is made of q-ary SR stages and GF(q) gates.

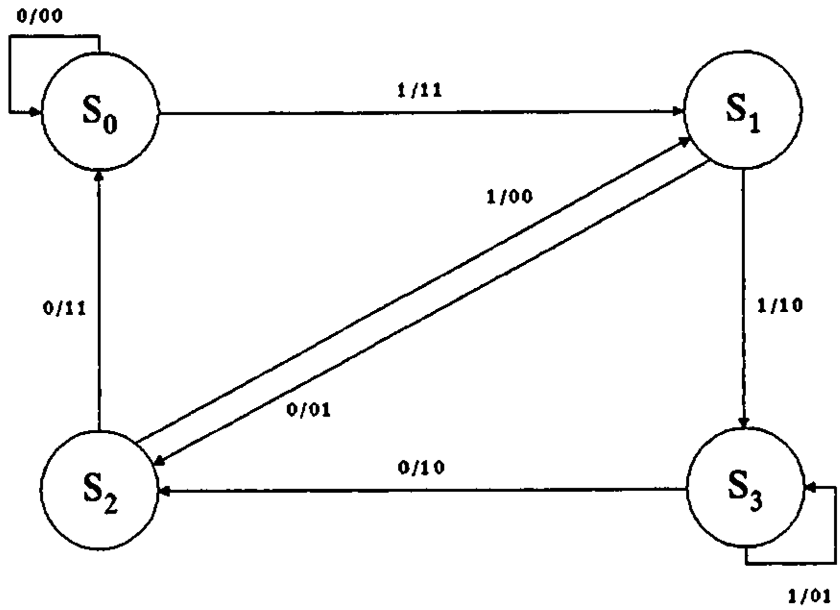


Figure A3.3.1: State-transition diagram for a (2,1,2) normal encoder.

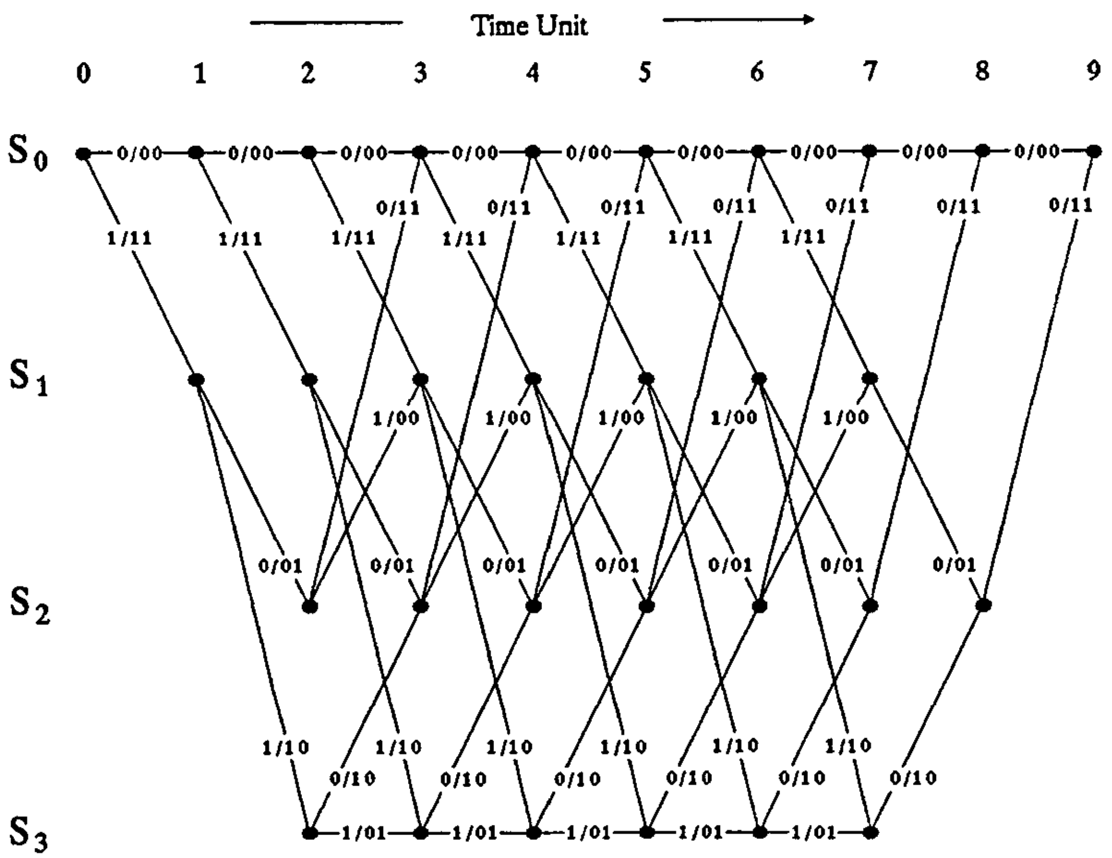


Figure A3.3.2: Trellis diagram for the (2,1,2) normal encoder with the state-transition diagram of Fig. A3.3.1.

During the remaining time-units (8 & 9), the encoder is reset (i.e. only zero-I/P is permitted).

∎

APPENDIX 3.4: EXAMPLE OF VITERBI DECODING

Example A3.4.1: Consider the normal encoder for the (2,1,2) code defined in Example A3.3.1 (p. 335). Its trellis diagram is shown in Fig. A3.3.2 (p. 336). Assume that transmission is over the binary symmetric channel. Consider the source sequence **u** = (0 1 0 0 1 1 1) made of L = 7 I/P k-tuples (here, k = 1). This is appended with mk 0s (to reset the encoder - here mk = 2) producing the channel sequence v

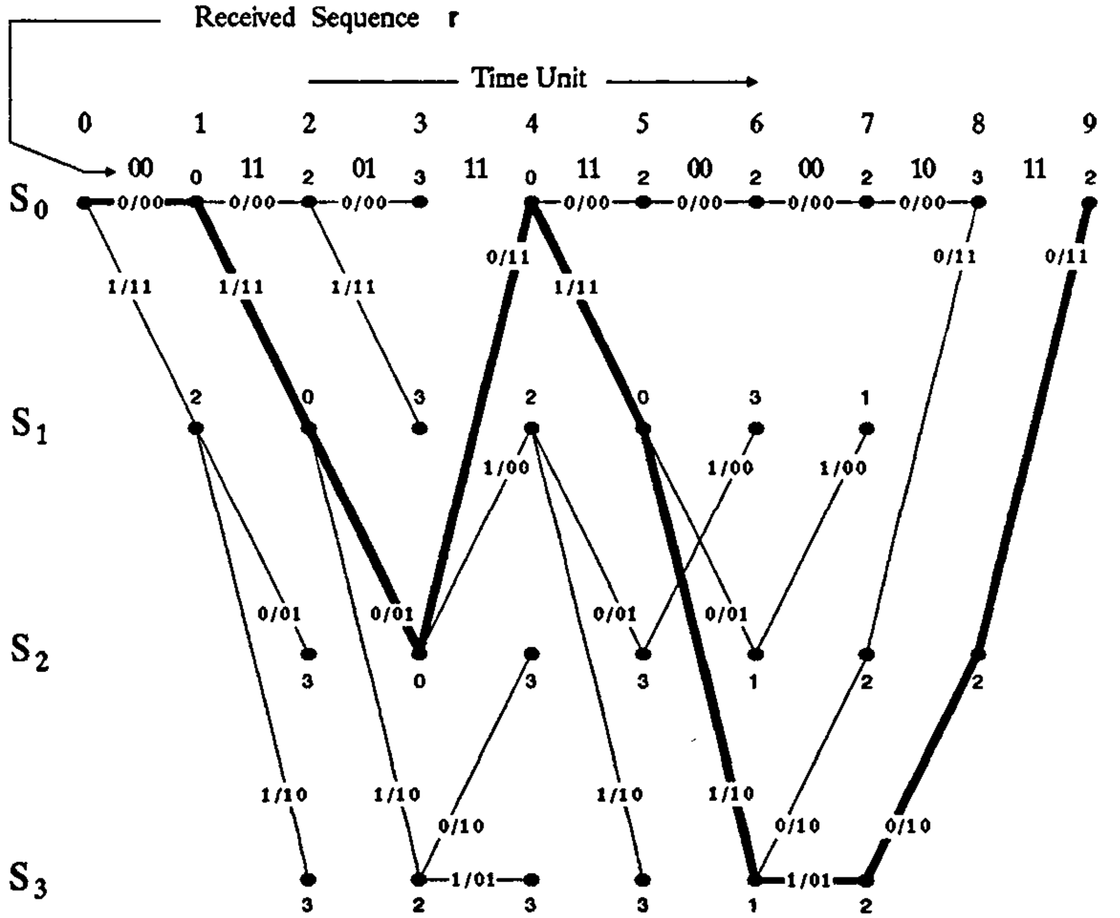


Figure A3.4.1: Example of Viterbi decoding using the encoder with the trellis diagram of Fig. A3.3.2 (p. 336) and assuming transmission over the binary symmetric channel.

= (0 0, 1 1, 0 1, 1 1, 1 1, 1 0, 0 1, 1 0, 1 1).

Let the channel error sequence e = (0 0, 0 0, 0 0, 0 0, 0 0, 1 0, 0 1, 0 0, 0 0), giving rise to the received sequence r = (0 0, 1 1, 0 1, 1 1, 1 1, 0 0, 0 0, 1 0, 1 1). This is used by the Viterbi decoder to produce its best estimate of the transmitted sequence v, which in this case coincides with the original; in other words, the decoder successfully corrected the two errors.

Note that at each time-unit, there is one survivor per state. For instance, at time-unit 5, the following paths have survived:

Corresponding to S_0, (0 1 0 0 0) with metric 2.
Corresponding to S_1, (0 1 0 0 1) with metric 0.
Corresponding to S_2, (0 1 0 1 0) with metric 3.
Corresponding to S_3, (0 1 0 1 1) with metric 3.

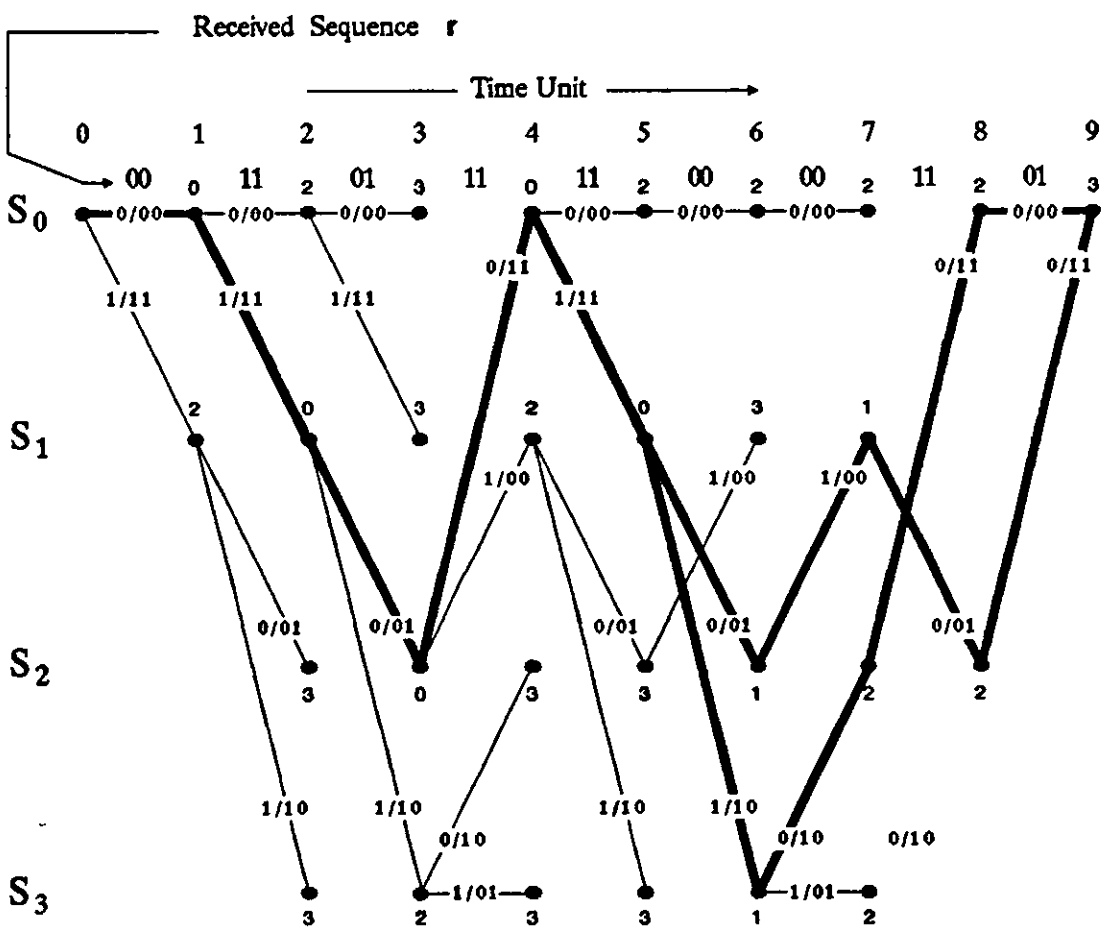


Figure A3.4.2: Example of Viterbi decoding, similar to the one in Fig. A3.4.1, but with four channel errors.

The final survivor (0 1 0 0 1 1 1 0 0) has metric 2 (i.e. it corresponds to a received sequence which is assumed to have been corrupted by two channel errors).

On occasion, a *tie* occurs, i.e. more than one paths entering a state have the same metric. In the case of a final tie, one path is arbitrarily selected. See for example in Fig. A3.4.2 a decoding case similar to the current one, but with four (instead of two) channel errors.

Note that there is a tie at time-unit 9. This gives rise to two final survivors, of which one has to be chosen:

u = (0 1 0 0 1 1 1 0 0)	
v = (0 0, 1 1, 0 1, 1 1, 1 1, 1 0, 0 1, 1 0, 1 1)	
ṽ = (0 0, 1 1, 0 1, 1 1, 1 1, 1 0, 1 0, 1 1, 0 0)	from #1
ṽ = (0 0, 1 1, 0 1, 1 1, 1 1, 0 1, 0 0, 0 1, 1 1)	from #2
e = (0 0, 0 0, 0 0, 0 0, 0 0, 1 0, 0 1, 0 1, 1 0)	
ẽ = (0 0, 0 0, 0 0, 0 0, 0 0, 0 0, 1 1, 0 1, 1 1)	from #1
ẽ = (0 0, 0 0, 0 0, 0 0, 0 0, 1 1, 0 1, 1 1, 0 0)	from #2
ũ = (0 1 0 0 1 1 0 0 0)	from #1
ũ = (0 1 0 0 1 0 1 0 0)	from #2

Note that the last two bits of **u** & **ũ** are used to reset the encoder. Comparing **u** & **ũ** one sees that four channel errors, in a span of 14 channel bits, are too much for this code, hence there is a single-bit decoding error.

∎

APPENDIX 3.5: <u>SEQUENTIAL DECODING</u>

Viterbi decoding has two important disadvantages: i) It can be used only with short constraint-length convolutional codes, because of limitations in the total encoder memory, M. ii) The number of computations per decoded source block is independent of the channel conditions (i.e. even if the channel is noiseless, the Viterbi decoder will do the same amount of computing). The number of computations of a sequential decoder on the other hand, depends on the noise level of the channel and is essentially independent of M. As a result, long constraint-length codes can be used and so arbitrarily low probabilities of (decoding) error can be achieved.

Wozencraft [30] introduced sequential decoding in the late fifties. In 1963, Fano [31] introduced a new sequential decoding algorithm and a few years later Zigangirov [32] (in 1966) and Jelinek [33] (independently, in 1969) introduced the third version, known as ZJ or stack algorithm. The easiest to use is the ZJ algorithm, while the Fano one is the most popular (see Clark & Cain [13], p. 298).

A sequential decoder does not follow the optimum procedure of the Viterbi decoder; instead of delaying decisions until some late level (time-unit), it looks at the first received block, makes a decision (based on some suitable metric) and proceeds to a point (state) at the next level. This procedure is repeated.

At each level the decoder is at a single node; since q^{k-f} q^f-label branches (see Theorem 3.2, p. 60) emerge from this node, the decoder chooses the one 'closest' to the received information and follows it to the next level. Any channel error will deflect the decoder from the correct path, but this will become apparent later on, because of the abnormal accumulation of errors; the decoder then backs-up, level by level, until it finds a suitable path and goes on as before.

Due to memory limitations the decoder can process only a finite portion of the trellis, say b levels at a time (the same applies to Viterbi decoding). Hence, at every level it makes irrevocable decisions about an old information block. The main disadvantage of sequential decoding is that the processing time per decoded block is a random variable with high deviation from the average; this can cause buffer overflow.

The *Fano algorithm* uses the *tilted distance*, $t(L)$, to detect incorrect paths. In particular, $t(L) = p'nL-d(L)$, where p' is a channel parameter (usually slightly larger than the channel error-rate, p) determined by simulation, L is the level (time-unit) and $d(L)$ is the Hamming distance between the received sequence and the current path through the trellis. If decoding is correct, $d(L) \approx pnL$, hence $t(L) \approx$

$nL(p'-p) > 0$ and increasing. If the tilted distance starts to decrease, the decoder backs-up.

The Fano algorithm is more time-consuming, while the ZJ one is more memory-consuming (see Lin & Costello [2], p. 360).

APPENDIX 3.6: TABLE LOOK-UP DECODING

This type of decoder has been employed where a very simple implementation is desired and only one to two dB coding gain* is required. Usually, hard-decision demodulator outputs are used and moderate constraint-lengths are necessary, to keep the size of the decoding table manageable [13].

The decoder is based on a table which relates syndrome patterns with channel error patterns.

The pairs are chosen so that the probability of decoding error, for the given channel, is minimized.

* See Section 1.4 (p. 13).

APPENDIX 4.1: PROOF OF THE THEORY IN SECTION 4.1

A4.1.1. Proof of Lemma 4.1

According to Lemma A2.10.1, the generator-polynomial matrix G(D) of an (n,k,m) non-catastrophic convolutional code has rank k. By means of elementary row operations (see Definition A2.2.6) G(D) can be transformed into one of its row-equivalents. If this row-equivalent is $[I_k,0]$, it is called a normal form (see Definition A2.2.8). Note that both matrices have the same rank (by Theorem A2.2.6). Then, by Theorem A2.2.7, there exist two nonsingular matrices A(D) & B(D) such that $G(D) = A(D)[I_k,0]B(D)$.

QED

A4.1.2. Proof of Theorem 4.1

According to Definition 2.12 (p. 44), the parity-check polynomial matrix associated with the k × n generator-polynomial matrix G(D) is any full-rank (n-k) × n matrix of polynomials which satisfies $G(D)H^T(D) = 0$.

The rank of $Y_2^T(D)$ is n-k because, from Note 4.1:

$$B(D)B^{-1}(D) = I_n \Longrightarrow$$

$$\begin{bmatrix} X_1(D)Y_1(D) & X_1(D)Y_2(D) \\ X_2(D)Y_1(D) & X_2(D)Y_2(D) \end{bmatrix} = \begin{bmatrix} I_k & 0 \\ 0 & I_{n-k} \end{bmatrix} \qquad (A4.1.1)$$

From (A4.1.1): $X_2(D)Y_2(D) = I_{n-k} \Longleftrightarrow Y_2^T(D)X_2^T(D) = I_{n-k}$

Hence, $Y_2^T(D)$ is an (n-k) × n matrix which has a right-inverse. So, it has rank = n-k < n (by Theorem A2.2.11).

Consider the product $G(D)[Y_2^T(D)]^T = G(D)Y_2(D)$. Using the Smith normal form and the partition of B(D) (see Lemma 4.1 & Note 4.1, p. 76),

$$G(D)Y_2(D) = A(D)\left[I_k, 0\right]\begin{bmatrix} X_1(D) \\ X_2(D) \end{bmatrix} Y_2(D)$$

$$\Longrightarrow \quad G(D)Y_2(D) = A(D)X_1(D)Y_2(D) = 0 \qquad \text{[by eqn (A4.1.1)]}$$

QED

A4.1.3. Proof of Theorem 4.2

With the help of Theorem 4.1, the syndrome equation can be written as

$$S(D) = E(D)Y_2(D) \tag{A}$$

Also, from eqn (4.2):

$$B^{-1}(D)\begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix} = \left[Y_1(D), Y_2(D)\right]\begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix} = Y_2(D) \tag{B}$$

$$S(D) = E(D)B^{-1}(D)\begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix}$$

Let $\quad W(D) \triangleq E(D)B^{-1}(D)$

$$\Longrightarrow \quad \Longleftrightarrow$$

$$S(D) = W(D)\begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix}$$

$$E(D) = W(D)B(D)$$

$$\Longrightarrow \quad \Longleftrightarrow$$

$$S(D) = \left[w^{(1)}(D), w^{(2)}(D), \ldots, w^{(n)}(D)\right]\begin{bmatrix} 0 \\ I_{n-k} \end{bmatrix}$$

$$E(D) = \left[w^{(1)}(D), w^{(2)}(D), \ldots, w^{(n)}(D)\right]\begin{bmatrix} X_1(D) \\ X_2(D) \end{bmatrix}$$

$$\Longrightarrow \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad \Longleftarrow \begin{cases} S(D) = \left[w^{(k+1)}(D), w^{(k+2)}(D), \ldots, w^{(n)}(D)\right] & \text{(C)} \\ E(D) = \left[w^{(1)}(D), w^{(2)}(D), \ldots, w^{(k)}(D)\right]X_1(D) & \\ \quad + \left[w^{(k+1)}(D), w^{(k+2)}(D), \ldots, w^{(n)}(D)\right]X_2(D) & \text{(D)} \end{cases}$$

Since $[w^{(1)}(D), w^{(2)}(D), \ldots, w^{(k)}(D)]$ is arbitrary, one can let

$$T(D) \triangleq \left[w^{(1)}(D), w^{(2)}(D), \ldots, w^{(k)}(D)\right] \qquad (E)$$

Note though that, from the definition of $W(D)$ above and the partition of $B^{-1}(D)$ (Note 4.1, p. 76),

$$W(D) = E(D)B^{-1}(D) = \left[E(D)Y_1(D), E(D)Y_2(D)\right] \Longrightarrow$$

$$\Longrightarrow \quad T(D) = E(D)Y_1(D) \qquad (F)$$

From eqns (C), (D) & (E): $E(D) = T(D)X_1(D) + S(D)X_2(D)$.

QED

A4.1.4. Proof of Theorem 4.3

Eqn (4.5), the result of Theorem 4.3, can be obtained from the result of Theorem 4.2 [eqn (4.4)] by providing expressions for $X_1(D)$ & $X_2(D)$.

From eqns (A4.1.1) & (4.3),

$$X_2(D)Y_2(D) = X_2(D)H^T(D) = I_{n-k} \Longrightarrow$$

$$\Longrightarrow \quad H(D)X_2^T(D) = I_{n-k}$$

$\Longrightarrow$ $X_2^T(D)$ is the right-inverse of $H(D)$, denoted by $H'(D)$:

$$X_2(D) = H'^T(D) \qquad (A4.1.2)$$

From eqns (4.1) & (4.2),

$$A^{-1}(D)G(D) = \left[I_k, 0\right]\begin{bmatrix} X_1(D) \\ X_2(D) \end{bmatrix} = X_1(D) \qquad (A4.1.3)$$

Substituting eqns (A4.1.2) & (A4.1.3) into eqn (4.4),

$$E(D) = T(D)A^{-1}(D)G(D) + S(D)H'^T(D)$$

By Theorem 4.2, $T(D)$ is a $1 \times k$ matrix. Then,

$$Z(D) \triangleq T(D)A^{-1}(D) \qquad (A4.1.4)$$

and from the last two eqns the final result follows.

QED

A4.1.5. Proof of Lemma 4.2

The result of Lemma 4.2 follows easily from eqn (4.5) and eqn (2.71) [the definition of S(D)]:

$$E(D) = Z(D)G(D) + R(D)H^T(D)H'^T(D)$$

Substituting $H^T(D) = Y_2(D)$ [from eqn (4.3)] and $H'^T(D) = X_2(D)$ [from eqn (A4.1.2)] the result of Lemma 4.2 is obtained.

QED

A4.1.6. Proof of Theorem 4.4

For systematic convolutional codes (see Lemma 2.9, p. 39) $G(D) = [I_k, P(D)]$ and using the Smith normal form (see Lemma 4.1) and the partition of B(D) (see Note 4.1),

$$A(D)\left[I_k, 0\right]\begin{bmatrix} X_1(D) \\ X_2(D) \end{bmatrix} = \left[I_k, P(D)\right] \longrightarrow$$

$$\longrightarrow \quad X_1(D) = A^{-1}(D)\left[I_k, P(D)\right] \qquad \text{(A4.1.5)}$$

From eqns (2.54) & (4.3), letting $Y_2^T(D) \triangleq \left[Y_{21}^T(D), Y_{22}^T(D)\right]$:

$$\left[I_k, P(D)\right]\begin{bmatrix} Y_{21}(D) \\ Y_{22}(D) \end{bmatrix} = 0 \quad \Longleftrightarrow \quad Y_{21}(D) = -P(D)Y_{22}(D)$$

$$\longrightarrow \quad Y_2(D) = \begin{bmatrix} -P(D) \\ I_{n-k} \end{bmatrix} Y_{22}(D) \qquad \text{(A4.1.6)}$$

Let:

$$X_2(D) \triangleq \left[X_{21}(D), X_{22}(D)\right] \quad \& \quad Y_1(D) \triangleq \begin{bmatrix} Y_{11}(D) \\ Y_{12}(D) \end{bmatrix}$$

where $X_{22}(D)$ & $Y_{11}(D)$ are square submatrices.

From (A4.1.1) & (A4.1.5):

$$X_1(D)Y_1(D) = I_k \Longrightarrow A^{-1}(D)\left[I_k, P(D)\right]\begin{bmatrix} Y_{11}(D) \\ Y_{12}(D) \end{bmatrix} = I_k \Longleftrightarrow$$

$$\Longrightarrow A^{-1}(D)Y_{11}(D) + A^{-1}(D)P(D)Y_{12}(D) = I_k \qquad (A)$$

From (A4.1.1):

$$X_2(D)Y_1(D) = 0 \Longrightarrow \left[X_{21}(D), X_{22}(D)\right]\begin{bmatrix} Y_{11}(D) \\ Y_{12}(D) \end{bmatrix} = 0 \Longleftrightarrow$$

$$\Longleftrightarrow X_{21}(D)Y_{11}(D) + X_{22}(D)Y_{12}(D) = 0 \qquad (B)$$

From (A4.1.1) & (A4.1.6):

$$X_2(D)Y_2(D) = I_{n-k} \Longrightarrow \left[X_{21}(D), X_{22}(D)\right]\begin{bmatrix} -P(D) \\ I_{n-k} \end{bmatrix} Y_{22}(D) = I_{n-k}$$

$$\Longrightarrow -X_{21}(D)P(D)Y_{22}(D) + X_{22}(D)Y_{22}(D) = I_{n-k} \qquad (C)$$

The next step is to solve the above system of matrix eqns, for the submatrices of $B(D)$ & $B^{-1}(D)$. Of the eight submatrices, $X_{11}(D)$, $X_{12}(D)$, $X_{21}(D)$, $X_{22}(D)$, $Y_{11}(D)$, $Y_{12}(D)$, $Y_{21}(D)$ & $Y_{22}(D)$, three are already known: $X_{11}(D)$ & $X_{12}(D)$, from (A4.1.5) and $Y_{21}(D)$, from (A4.1.6). With three eqns available it is obvious that two of the submatrices should be arbitrary.

$X_{21}(D)$ is a submatrix of $X_2(D)$, which is $(n-k) \times n$ (see Note 4.1, p. 76), with $X_{22}(D)$ being square, hence $(n-k) \times (n-k)$. So, $X_{21}(D)$ is an arbitrary $(n-k) \times k$ matrix:

$$X_{21}(D) \triangleq C(D) \qquad (D)$$

From Note 4.1 (p. 76), $Y_2(D)$ is $n \times (n-k)$. Hence, $Y_{22}(D)$ is $(n-k) \times (n-k)$ [see also (A4.1.6)] and so its rank cannot exceed $n-k$ (see Definition A2.2.7, p. 302). On the other hand $Y_2^T(D) = H(D)$ (by Theorem 4.1, p. 76), so $Y_2^T(D)$ has rank $n-k$. From (A4.1.6), $Y_2^T(D)$ is the product of two matrices non of which may have rank less than $n-k$ (by Theorem A2.2.15, p. 303). So the rank of the square matrix $Y_{22}^T(D)$ is $n-k$ and, by Theorem A2.2.12 (p. 303), this matrix is nonsingular. Hence, $Y_{22}^T(D)$ is an $(n-k) \times (n-k)$ nonsingular matrix and so is its transpose, $Y_{22}(D)$ [see Theorem A2.2.3 (iii), p. 300]:

$$Y_{22}(D) \triangleq F(D) \qquad (E)$$

From eqns (A) - (E), and since A(D) & F(D) are nonsingular:

$$Y_{11}(D) = A(D) - P(D)Y_{12}(D) \qquad (F)$$

$$C(D)Y_{11}(D) = -X_{22}(D)Y_{12}(D) \qquad (G)$$

$$X_{22}(D) = F^{-1}(D) + C(D)P(D) \qquad (H)$$

Substituting (F) & (H) in (G):

$$C(D)A(D) - C(D)P(D)Y_{12}(D) = -F^{-1}(D)Y_{12}(D) - C(D)P(D)Y_{12}(D)$$

$$\Longleftrightarrow \quad Y_{12}(D) = -F(D)C(D)A(D) \qquad (I)$$

Substituting (I) in (F):

$$Y_{11}(D) = A(D) + P(D)F(D)C(D)A(D) \qquad (J)$$

From Note 4.1 (p. 76), (A4.1.5), (D) & (H), B(D) can be pieced togeher:

$$B(D) = \begin{bmatrix} A^{-1}(D) & A^{-1}(D)P(D) \\ C(D) & F^{-1}(D) + C(D)P(D) \end{bmatrix} \qquad (4.7a)$$

From Note 4.1 (p. 76), (A4.1.6), (E), (I) & (J), $B^{-1}(D)$ can be pieced togeher:

$$B^{-1}(D) = \begin{bmatrix} A(D) + P(D)F(D)C(D)A(D) & -P(D)F(D) \\ -F(D)C(D)A(D) & F(D) \end{bmatrix} \qquad (4.7b)$$

QED

A4.1.7. Proof of Theorem 4.5

From eqn (4.4): $E(D) = T(D)X_1(D) + S(D)X_2(D)$ (A)

From partition (4.2) and eqn (4.7a):

$$\begin{bmatrix} X_1(D) \\ X_2(D) \end{bmatrix} = \begin{bmatrix} A^{-1}(D) & A^{-1}(D)P(D) \\ C(D) & F^{-1}(D) + C(D)P(D) \end{bmatrix} \qquad (B)$$

From eqns (A) & (B):

$$E(D) = T(D)\left[A^{-1}(D), A^{-1}(D)P(D)\right] + S(D)\left[C(D), F^{-1}(D) + C(D)P(D)\right] \Longrightarrow$$

$$E(D) = \left[T(D)A^{-1}(D), T(D)A^{-1}(D)P(D)\right] +$$

$$+ \left[S(D)C(D), S(D)F^{-1}(D) + S(D)C(D)P(D)\right] =$$

$$= \left[T(D)A^{-1}(D) + S(D)C(D), T(D)A^{-1}(D)P(D) + S(D)F^{-1}(D) + S(D)C(D)P(D)\right]$$

Using $Z(D) \triangleq T(D)A^{-1}(D) + S(D)C(D)$ in the above eqn:

$$E(D) = \left[\; Z(D), Z(D)P(D) + S(D)F^{-1}(D)\right] \qquad (A4.1.7)$$

and since $T(D)$ is arbitrary, so is $T(D)A^{-1}(D) + S(D)C(D) = Z(D)$. The theorem is proved by letting $F(D) = I_{n-k}$ [recall, from § A4.1.6., that $F(D)$ is an arbitrary nonsingular $(n-k) \times (n-k)$ matrix].

QED

A4.1.8. Proof of Theorem 4.6

From the basic eqn of the additive-noise channel [eqn (2.70)], $R(D) = V(D) + E(D)$. Then the best estimate, $\tilde{V}(D)$, of the channel sequence is (using Lemma 4.2, p. 77),

$$\tilde{V}(D) = R(D) - \tilde{E}(D) = R(D) - \left[\tilde{Z}(D)G(D) + R(D)Y_2(D)X_2(D)\right] \qquad (A)$$

Since $G(D)$ is a $k \times n$ matrix of rank k (see Lemma A2.10.1, p. 319), it has a right-inverse (see Theorem A2.2.11, p. 303) denoted by, say, $G'(D)$. From the Smith normal form (see Lemma 4.1, p. 76), it can be easily verified that,

$$\text{If } G(D)G'(D) = I_k \Longrightarrow G'(D) = B^{-1}(D)\begin{bmatrix} I_k \\ 0 \end{bmatrix}A^{-1}(D) \qquad (A4.1.8)$$

From the fundamental eqn $V(D) = U(D)G(D)$, post-multiplying with $G'(D)$, $V(D)G'(D) = U(D)$ and using (A),

$$\tilde{U}(D) = R(D)G'(D) - \tilde{Z}(D)G(D)G'(D) - R(D)Y_2(D)X_2(D)G'(D)$$

$$\Longrightarrow \tilde{U}(D) = R(D)G'(D) - \tilde{Z}(D) - R(D)Y_2(D)X_2(D)G'(D) \qquad (B)$$

From eqns (A4.1.8) & (4.2),

$$Y_2(D)X_2(D)G'(D) = Y_2(D)X_2(D)\left[Y_1(D), Y_2(D)\right]\begin{bmatrix} I_k \\ 0 \end{bmatrix}A^{-1}(D)$$

$$\Longrightarrow \quad Y_2(D)X_2(D)G'(D) = Y_2(D)X_2(D)Y_1(D)A^{-1}(D)$$

$$\Longrightarrow \quad Y_2(D)X_2(D)G'(D) = Y_2(D)0A^{-1}(D) = 0 \quad \text{[by eqn (A4.1.1)]}$$

From the last result & eqn (B), Theorem 4.6 is proved.

QED

A4.1.9. Proof of Theorem 4.7

Substituting $B^{-1}(D)$ (from Theorem 4.4, p. 78) in the expression for the right-inverse, $G'(D)$, of $G(D)$ [see eqn (A4.1.8)]:

$$G'(D) = \begin{bmatrix} A(D) + P(D)F(D)C(D)A(D) & -P(D)F(D) \\ -F(D)C(D)A(D) & F(D) \end{bmatrix}\begin{bmatrix} A^{-1}(D) \\ 0 \end{bmatrix}$$

$$\Longrightarrow \quad G'(D) = \begin{bmatrix} I_k + P(D)F(D)C(D) \\ -F(D)C(D) \end{bmatrix} \qquad \text{(A4.1.9)}$$

Also, since the right-inverse of $G(D)$ exists, eqn (2.41a) (p. 33) can be inverted to give

$\tilde{U}(D) = \tilde{V}(D)G'(D)$, or using eqn (A4.1.9),

$$\tilde{U}(D) = \tilde{V}(D)\begin{bmatrix} I_k + P(D)F(D)C(D) \\ -F(D)C(D) \end{bmatrix} \qquad \text{(A)}$$

From eqn (2.70) $\tilde{V}(D) = R(D) - \tilde{E}(D)$ and combining with eqn (A)

$$\tilde{U}(D) = \left[R(D) - \tilde{E}(D)\right]\begin{bmatrix} I_k + P(D)F(D)C(D) \\ -F(D)C(D) \end{bmatrix} \qquad \text{(B)}$$

From Lemma 2.11: $$R(D) = \left[R^{(m)}(D), R^{(p)}(D)\right] \qquad \text{(C)}$$

From eqn (A4.1.7): $$\tilde{E}(D) = \left[\tilde{Z}(D), \tilde{Z}(D)P(D)+S(D)F^{-1}(D)\right] \qquad \text{(D)}$$

From eqns (B), (C) & (D), $\tilde{U}(D) =$

$$= \left[R^{(m)}(D)-\tilde{Z}(D), R^{(p)}(D)-\tilde{Z}(D)P(D)-S(D)F^{-1}(D)\right]\begin{bmatrix} I_k + P(D)F(D)C(D) \\ -F(D)C(D) \end{bmatrix}$$

$$\longrightarrow \quad \tilde{U}(D) = R^{(m)}(D) - \tilde{Z}(D) + R^{(m)}(D)P(D)F(D)C(D) -$$

$$- \tilde{Z}(D)P(D)F(D)C(D) - R^{(p)}(D)F(D)C(D) + \tilde{Z}(D)P(D)F(D)C(D) +$$

$$+ S(D)F^{-1}(D)F(D)C(D)$$

$$\longrightarrow \quad \tilde{U}(D) = R^{(m)}(D) - \tilde{Z}(D) + S(D)C(D) +$$

$$+ \left[R^{(m)}(D)P(D) - R^{(p)}(D)\right]F(D)C(D) \qquad (E)$$

From eqn (2.71) (the definition of S(D), p. 48) and eqn (4.3) (p. 76), $S(D) \triangleq R(D)H^T(D) = R(D)Y_2(D)$. From Theorem 4.4 (p. 78) the general expression for $Y_2(D)$ is obtained, while from Lemma 2.11 (p. 48) the partition of R(D) is used:

$$S(D) = \left[R^{(m)}(D), R^{(p)}(D)\right]\begin{bmatrix} -P(D)F(D) \\ F(D) \end{bmatrix} \quad \longrightarrow$$

$$\longrightarrow \quad S(D) = -R^{(m)}(D)P(D)F(D) + R^{(p)}(D)F(D) \qquad (F)$$

From eqns (E) & (F):

$$\tilde{U}(D) = R^{(m)}(D) - \tilde{Z}(D) - \left[R^{(m)}(D)P(D)F(D) - R^{(p)}(D)F(D)\right]C(D) +$$

$$+ \left[R^{(m)}(D)P(D)F(D) - R^{(p)}(D)F(D)\right]C(D) \quad \longrightarrow$$

$$\longrightarrow \quad \tilde{U}(D) = R^{(m)}(D) - \tilde{Z}(D)$$

QED

APPENDIX 4.2: SET THEORY AND PARTITIONS

The basic operations on sets are the operations of *union* and *intersection* (which are assumed to be well known). A third operation is introduced below:

Definition A4.2.1: Let A & B be any two sets. Then, the *relative complement of* B *in* A is denoted by A-B and is defined to be the set of all the elements of A that do not belong to B:

$$A-B \mathrel{\hat{=}} \{x \;/\; x\in A \;\&\; x\notin B\} \qquad (A4.2.1)$$

∎

<u>Definition A4.2.2</u> In many applications, all sets are subsets of a large set which is called the *universal set* and is denoted by S. The *complement* of B can be defined to be the set S-B which, by Definition A4.2.1, is

$$-B \mathrel{\hat{=}} S-B = \{x \;/\; x\in S \;\&\; x\notin B\} \qquad (A4.2.2)$$

∎

The following set-theory identities can be found in any set-theory chapter or book (see for example Enderton [34]):

<u>Note A4.2.1</u>: "The following identities, which hold true for any sets, are some of the elementary facts of the algebra of sets" [34]:

Commutative laws:

$$A \cup B = B \cup A \quad \text{and} \quad A \cap B = B \cap A \qquad (A4.2.3)$$

Associative laws:

$$A \cup (B \cup C) = (A \cup B) \cup C \qquad (A4.2.4a)$$

$$A \cap (B \cap C) = (A \cap B) \cap C \qquad (A4.2.4b)$$

Distributive laws:

$$A \cap (B \cup C) = (A \cap B) \cup (A \cap C) \qquad (A4.2.5a)$$

$$A \cup (B \cap C) = (A \cup B) \cap (A \cup C) \qquad (A4.2.5b)$$

De Morgan's laws:

$$C - (A \cup B) = (C - A) \cap (C - B) \qquad (A4.2.6a)$$

$$C - (A \cap B) = (C - A) \cup (C - B) \qquad (A4.2.6b)$$

If C = S:

$$-(A \cup B) = -A \cap -B \qquad (A4.2.6c)$$

$$-(A \cap B) = -A \cup -B \qquad (A4.2.6d)$$

Identities involving the empty set, ∅:

$$A \cup \emptyset = A \quad \& \quad A \cap \emptyset = \emptyset \quad \& \quad A \cap (C - A) = \emptyset \qquad (A4.2.7)$$

Identities involving the universal set, S, (if $A \subseteq S$):*

$$A \cup S = S \quad \& \quad A \cap S = A \quad \& \quad A \cup -A = S \qquad (A4.2.8)$$

∎

<u>Definition A4.2.3</u>: Non-empty sets $X_1, X_2, \ldots, X_n$ are said to *partition a set* Y, i.e. $Y = \langle X_1, X_2, \ldots, X_n \rangle$, if, and only

* Remember, $A \subseteq B$ denotes "A is a *subset* of B".

if, their union equals Y and their intersection in pairs is the empty set:

$$\left.\begin{array}{r} \bigcup_{i=1}^{n} X_i = Y \\ X_i \cap X_j = \emptyset \quad /\text{for all } i,j \in [1,n] : i \neq j \end{array}\right] \longrightarrow \qquad (A4.2.9)$$

■

Theorem A4.2.1: Let A,B be any two non-empty subsets of the universal set, S. Then:

$$A \cup B = \langle A, B-A \rangle \qquad (A4.2.10a)$$

$$\text{If } A \subset B \Longrightarrow B = \langle A, B-A \rangle \qquad (A4.2.10b)$$

Proof: From Definition A4.2.1, one may write:

$$B-A = \{x \;/\; x \in B \;\&\; x \notin A\} = B \cap -A \qquad (A4.2.11)$$

Consider the set $A \cap (B-A)$ and use eqns (A4.2.11), (A4.2.3) (the commutative law), (A4.2.4b) (the associative law) and (A4.2.7):

$$A \cap (B-A) = A \cap (B \cap -A) = A \cap (-A \cap B) \Longrightarrow$$

$$A \cap (B-A) = (A \cap -A) \cap B = \emptyset \cap B = \emptyset \qquad (A)$$

Consider the set $A \cup (B-A)$ and use eqns (A4.2.11), (A4.2.5b) (distributive law) and (A4.2.8):

$$A \cup (B-A) = A \cup (B \cap -A) = (A \cup B) \cap (A \cup -A) \Longrightarrow$$

$$A \cup (B-A) = (A \cup B) \cap S = A \cup B \qquad (B)$$

It is clear from eqns (A) & (B) and Definition A4.2.3 that A & B-A partition $A \cup B$ (A,B are of course non-empty). It is also clear that if $A \subset B$ then $A \cup B = B$.

QED

Theorem A4.2.2: Let A, B & C be any two subsets of the universal set S. Then:

$$(C-A) - (B-A) = C - A \cup B \qquad (A4.2.12a)$$

$$-A - (B-A) = -A \cup B \qquad (A4.2.12b)$$

$$\text{If } A \subseteq B \text{ then} \quad -A - (B-A) = -B \qquad \text{(A4.2.12c)}$$

Proof: Let $X = (C-A) - (B-A)$. From eqns (A4.2.11) & (A4.2.6d):

$$X = (C \cap -A) \cap [-(B \cap -A)] = (C \cap -A) \cap (-B \cup A)$$

$$\Longrightarrow \quad X = [(C \cap -A) \cap -B] \cup [(C \cap -A) \cap A] \quad \text{[by (A4.2.5a)]}$$

$$\Longrightarrow \quad X = [(C \cap -A) \cap -B] \cup [C \cap (-A \cap A)] = (C \cap -A) \cap -B$$

$$\Longrightarrow \quad X = C \cap (-A \cap -B) = C \cap -(A \cup B) \quad \text{[by (A4.2.6c)]}$$

$$\Longrightarrow \quad X = (C-A) - (B-A) = C - A \cup B \qquad \text{(A)}$$

If $C = S$ in eqn (A), result (b) is readily obtained.

If $A \subseteq B$, then $A \cup B = B$. This proves result (c).

QED

APPENDIX 4.3: PROOF OF THEOREMS 4.9 & 4.10

A4.3.1. Proof of Theorem 4.9

Since $A_1, A_2, \ldots, A_\alpha$ partition B they are non-empty sets (see Definition A4.2.3). If $\beta(i) \triangleq |A_i|$ (= the number of elements of A_i), the A_is can be listed in the usual way:

$$\text{For all } i=1,2,\ldots,\alpha: \quad A_i = \left\{a_1^{(i)}, a_2^{(i)}, \ldots, a_{\beta(i)}^{(i)}\right\} \qquad \text{(A)}$$

Also, and for the same reason as above, they are mutually disjoint, hence their union can be listed in a similar way:

$$A_1 \cup A_2 \cup \cdots \cup A_\alpha = \left\{a_1^{(1)}, \ldots, a_{\beta(1)}^{(1)}, a_1^{(2)}, \ldots, a_{\beta(2)}^{(2)}, \ldots, a_1^{(\alpha)}, \ldots, a_{\beta(\alpha)}^{(\alpha)}\right\} \qquad \text{(B)}$$

Finally, and for the same reason as above, their union equals B, hence eqn (B) can be re-written:

$$B = \left\{a_1^{(1)}, \ldots, a_{\beta(1)}^{(1)}, a_1^{(2)}, \ldots, a_{\beta(2)}^{(2)}, \ldots, a_1^{(\alpha)}, \ldots, a_{\beta(\alpha)}^{(\alpha)}\right\} \qquad \text{(C)}$$

Then:

$$\beta(1) + \beta(2) + \cdots + \beta(\alpha) = |B| \qquad \text{(D)}$$

The state of (a part of) an LSC, at a time-unit h, is defined to be the $|B|$-tuple of the contents of its memory, that is, the ordered set of the $|B|$ bits that occupy its memory. The particular order used is arbitrary, hence of no importance for the current discussion; consequently, one may write

$$B(h) = \left[a_1^{(1)}a_2^{(1)}\cdots a_{\beta(1)}^{(1)}a_1^{(2)}a_2^{(2)}\cdots a_{\beta(2)}^{(2)}\cdots a_1^{(\alpha)}a_2^{(\alpha)}\cdots a_{\beta(\alpha)}^{(\alpha)}\right] \qquad (E)$$

$$\text{For all } i=1,2,\ldots,\alpha: \qquad A_i(h) = \left[a_1^{(i)}a_2^{(i)}\cdots a_{\beta(i)}^{(i)}\right] \qquad (F)$$

Since the LSCs are assumed to be binary, the Hamming weight of an x-tuple is simply the algebraic sum of its components:

$$\text{For all } i=1,2,..,\alpha: \quad w\left[A_i(h)\right] = a_1^{(i)} + a_2^{(i)} + \cdots + a_{\beta(i)}^{(i)} \qquad (G)$$

$$w\left[B(h)\right] = a_1^{(1)} + a_2^{(1)} + \cdots + a_{\beta(1)}^{(1)} + a_1^{(2)} + a_2^{(2)} + \cdots + a_{\beta(2)}^{(2)} + \cdots + a_1^{(\alpha)} + a_2^{(\alpha)} + \cdots + a_{\beta(\alpha)}^{(\alpha)} \qquad (H)$$

From eqns (G) & (H), the final result is obtained:

$$w\left[B(h)\right] = w\left[A_1(h)\right] + w\left[A_2(h)\right] + \cdots + w\left[A_\alpha(h)\right]$$

QED

A4.3.2. Proof of Theorem 4.10

By Definition 3.1 (p. 58), the memory group MEG of the regulator circuit is made of a collection of past I/P bits. Specifically, from eqn (3.5a):

$$MEG(h) = \left\{z_{h-1}^{(i)}, z_{h-2}^{(i)}, \ldots, z_{h-M_i}^{(i)} \;/ i=1,2,\ldots,k : M_i \geq 1\right\} \qquad (A)$$

Since $M_i \leq m$ for all $i=1,2,\ldots,k$ [see eqn (3.1)] then from eqn (A):

$$MEG(h) \subseteq \left\{z_{h-1}^{(i)}, z_{h-2}^{(i)}, \ldots, z_{h-m}^{(i)} \;/ i=1,2,\ldots,k\right\}^* \qquad \Longleftrightarrow$$

$$\left(MEG(h) \cup ING(h)\right) \subseteq \left\{z_h^{(i)}, z_{h-1}^{(i)}, \ldots, z_{h-m}^{(i)} \;/ i=1,2,\ldots,k\right\} \qquad \Longrightarrow$$

$$w\left[S(h) \cup z_h\right] \leq w\left[z_h^{(1)}\cdots z_h^{(k)}z_{h-1}^{(1)}\cdots z_{h-1}^{(k)}\cdots z_{h-m}^{(1)}\cdots z_{h-m}^{(k)}\right] \qquad (B)$$

Reln (B) follows from Lemma 4.6, noting that S(h) is the state of MEG and z_h is the state of ING, at time-unit h. From (B):

$$w\left[S(h) \cup z_h\right] \leq \sum_{i=0}^{m} w\left[z_{h-i}\right] \leq t \qquad \text{(by Theorem 4.8, p. 86)}$$

* Remember, A⊆B denotes "A is a *subset* of B".

Note that MEG(h) & ING(h) are disjoint sets by construction: ING(h) is made of the components of z_h, while MEG(h) is made of the components of $z_{h-1},\ldots,z_{h-m}$. Hence, MEG(h) $\cap$ ING(h) = $\emptyset$. Consequently, MEG(h) $\cup$ ING(h) = <MEG(h),ING(h)> (by Definition A4.2.3) hence, by Theorem 4.9:

$$w\left[S(h) \cup z_h\right] = w\left[S(h)\right] + w\left[z_h\right]$$

From the last two results: $w\left[z_h\right] \le t - w\left[S(h)\right]$

QED

APPENDIX 4.4: PROOF OF THEOREM 4.11

The above results are based on the fact that the number of combinations of k things, taken i at a time, is C(k,i) (see, for example, S. Lipschutz [35], Section 8.6).

Then, $\mathcal{E}(i)$ is the number of M-tuples of weight i and this is C(M,i). Furthermore, according to Lemma 4.7 (p. 88), the weight of a state may vary between 0 and t, inclusive.

Similarly, assume that the current state has weight w. Then, by Theorem 4.10 (p. 88), the weight i of the current input-block must not exceed t-w. Furthermore, there are C(k,i) k-tuples of weight i.

Finally, according to the above, if the current state has weight w, there are $\varrho(i,w)$ (where $0 \le i \le t-w$) different permitted input blocks, hence there are as many ways to change state. If all k SRs have non-zero length, then each new input block leads to a new unique state. If though, f SRs have zero length (i.e. they do not exist - see discussion in Section 3.2) then some of the input blocks lead to the same state. For example, if the 3rd row of G(D) contains only 'ones' or 'zeros' then the 3rd SR does not exist. Two I/P blocks that differ only in the 3rd bit will lead to the same state. Hence, the input-block bits that participate in a state change are k-f and there exist C(k-f,i) (k-f)-tuples of weight i.

QED

APPENDIX 4.5: EXAMPLE OF CONSTRAINED REGULATOR TRELLIS

Example A4.5.1: Consider the regulator circuit with transfer-function matrix:

$$P(D) = \begin{bmatrix} 1+D+D^2 \\ 1+D \\ 1+D^2 \end{bmatrix}$$

Obviously, n-k=1, k=3 & m=2, hence it is the transfer function of a (4,3,2) regulator circuit. The total memory is 5, i.e. the state-transition diagram has 32 states.

The diagram of the regulator circuit is shown in Fig. A4.5.1.

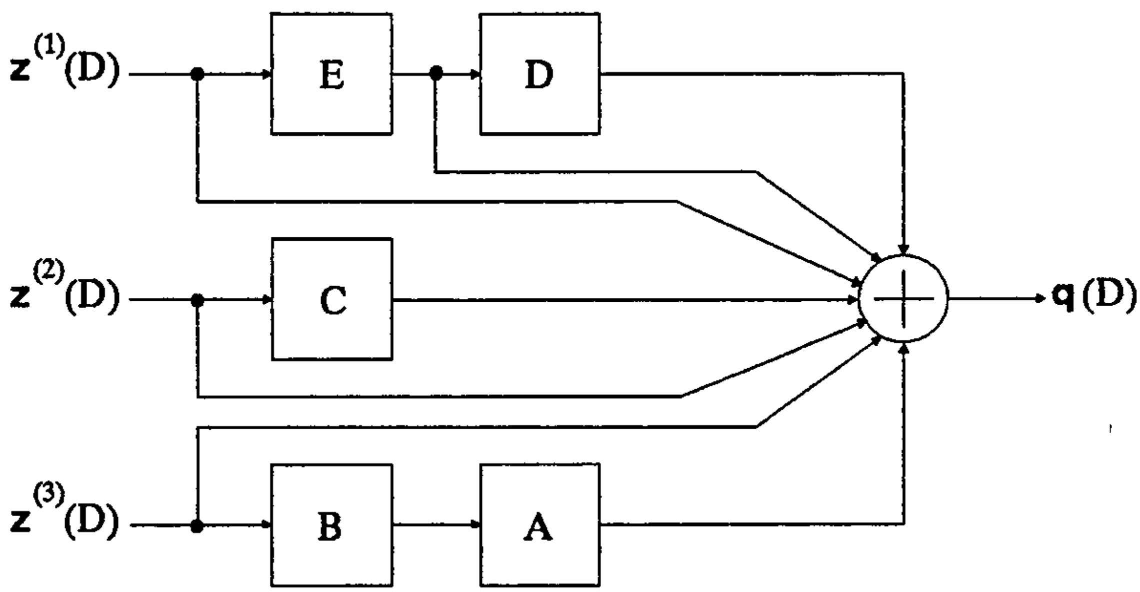


Figure A4.5.1: Circuit diagram of a (4,3,2) regulator circuit.

The free distance, d_{free}, of the associated code equals the weight of the minimum-weight codeword which is non-zero in its first block (see Appendix 2.5, p. 311 & Appendix 2.10, p. 317). d_{free} can be obtained from the trellis diagram, by finding that sequnce of output blocks which has minimum weight and is non-zero in its first block. Whatever this sequence, there will be a time-unit at which it will remerge with state S_0 and remain in the $S_0 \longrightarrow S_0$ transitions, which exist (by Lemma 3.3, p. 67) and which occur with an all-zero input and (hence) with an all-zero output (ibid.). So, d_{free} will assume a finite value at some point. Nevertheless, the trellis has 32 states, hence it is difficult to determine d_{free} this way.

From the transfer-function matrix, $v^{(i)}(D) = u^{(i)}(D)$ /i=1,2,3 and $v^{(4)}(D) = (1+D+D^2)u^{(1)}(D) + (1+D)u^{(2)}(D) + (1+D^2)u^{(3)}(D)$. Since $V(D) = [v^{(1)}(D), v^{(2)}(D), v^{(3)}(D), v^{(4)}(D)]$, the minimum-weight V(D) which is non-zero in its first block must contain at least one 1 and the minimum possible number of powers of D. Clearly, at least one of the $u^{(i)}(D)$s must contain one 1. If $u^{(2)}(D) = 1$ and $u^{(1)}(D) = u^{(3)}(D) = 0$, then $w[V(D)] = w[0,1,0,1+D] = 3$. The only way $w[V(D)] = w[u^{(1)}(D)] + w[u^{(2)}(D)] + w[u^{(3)}(D)] + w[v^{(4)}(D)] < 3$, is if the I/P contains one or two terms (of which one must be 1) and $v^{(4)}(D)$ contains one or no term, respectively.

If $u^{(1)}(D) = 1$ [and $u^{(2)}(D) = u^{(3)}(D) = 0$], then $V(D) = [1,0,0,1+D+D^2]$, while if $u^{(3)}(D) = 1$ [and $u^{(1)}(D) = u^{(2)}(D) = 0$], then $V(D) = [0,0,1,1+D^2]$.

If U(D) is to contain one 1 and another term, then $v^{(4)}(D)$ must be zero so that a free distance of 2 is obtained. In other words, $u^{(1)}(D)$, $u^{(2)}(D)$ and $u^{(3)}(D)$ must satisfy:

$$v^{(4)}(D) = (1+D+D^2)u^{(1)}(D) + (1+D)u^{(2)}(D) + (1+D^2)u^{(3)}(D) = 0$$

$$\Longleftrightarrow$$

$$[u^{(1)}(D) + u^{(2)}(D) + u^{(3)}(D)] + D[u^{(1)}(D) + u^{(2)}(D)] + D^2[u^{(1)}(D) + u^{(3)}(D)] = 0 \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad \left\{ \begin{array}{l} u^{(1)}(D) + u^{(2)}(D) + u^{(3)}(D) = 0 \\ u^{(1)}(D) + u^{(2)}(D) = 0 \\ u^{(1)}(D) + u^{(3)}(D) = 0 \end{array} \right.$$

$$\Longleftrightarrow \quad u^{(1)}(D) = u^{(2)}(D) = u^{(3)}(D) = 0$$

The last solution is not acceptable, hence $d_{free} = 3$. This agrees with Blahut [10] and Reed & Truong [24] (the latter use this code to illustrate their technique). Then, the error-correcting capability is t=1.

The transition diagram of the regulator circuit is constructed following the directions of Note 4.5 and the experience of the examples of Chapter 3.

Note here that Reed & Truong [24] (who use this example) talk about and use a regulator circuit with 6 SR stages: "...the number of internal states ... of the regulator circuit can be limited to seven out of a possible 64." This happens because they assume a circuit realization which uses 3 SRs each of length 2. Note also that the total memory of the regulator circuit could have been reduced to 2, hence giving rise to 4 states (for the unconstrained case), if a type-II realization was to be used. Nevertheless, in this case, all the preceding analysis wouldn't have been valid.

According to Lemma 4.7, the following states are permitted:

TABLE A4.5.1

A B C D E	S_j
0 0 0 0 0	S_0
0 0 0 0 1	S_1
0 0 0 1 0	S_2
0 0 1 0 0	S_3
0 1 0 0 0	S_4
1 0 0 0 0	S_5

The following results are easily obtained, from Fig. A4.5.1. To simplify notation, let $z^{(1)}(D) = z_1$, $z^{(2)}(D) = z_2$, $z^{(3)}(D) = z_3$, $q(D) = q$ and let the next state be S'. Then:

$$S = [ABCDE] \qquad S' = [Bz_3z_2Ez_1]$$

$$\text{and} \quad q = z_1 + z_2 + z_3 + A + C + D + E$$

If the current state is $S = S_0 = [00000]$, then the above eqns are simplified to:

$$S' = [0z_3z_20z_1] \qquad \text{and} \qquad q = z_1 + z_2 + z_3$$

According to Theorem 4.10 (p. 88), $w[S] + w[z_1,z_2,z_3] \le t = 1 \Longrightarrow w[z_1,z_2,z_3] = 0 \text{ or } 1$. Then:

TABLE A4.5.2

z_1	z_2	z_3	q	S'
0	0	0	0	S_0
0	0	1	1	S_4
0	1	0	1	S_3
1	0	0	1	S_1

If the current state has weight 1, then according to Theorem 4.10 (p. 88), $w[S] + w[z_1,z_2,z_3] \le t = 1 \Longrightarrow w[z_1,z_2,z_3] = 0 \Longrightarrow z_1 = z_2 = z_3 = 0$. The above eqns are simplified to:

$$S' = [B00E0] \quad \text{and} \quad q = A + C + D + E$$

TABLE A4.5.3

S = [ABCDE]	q	S' = [B00E0]
S_1 = [00001]	1	S_2 = [00010]
S_2 = [00010]	1	S_0 = [00000]
S_3 = [00100]	1	S_0 = [00000]
S_4 = [01000]	0	S_5 = [10000]
S_5 = [10000]	1	S_0 = [00000]

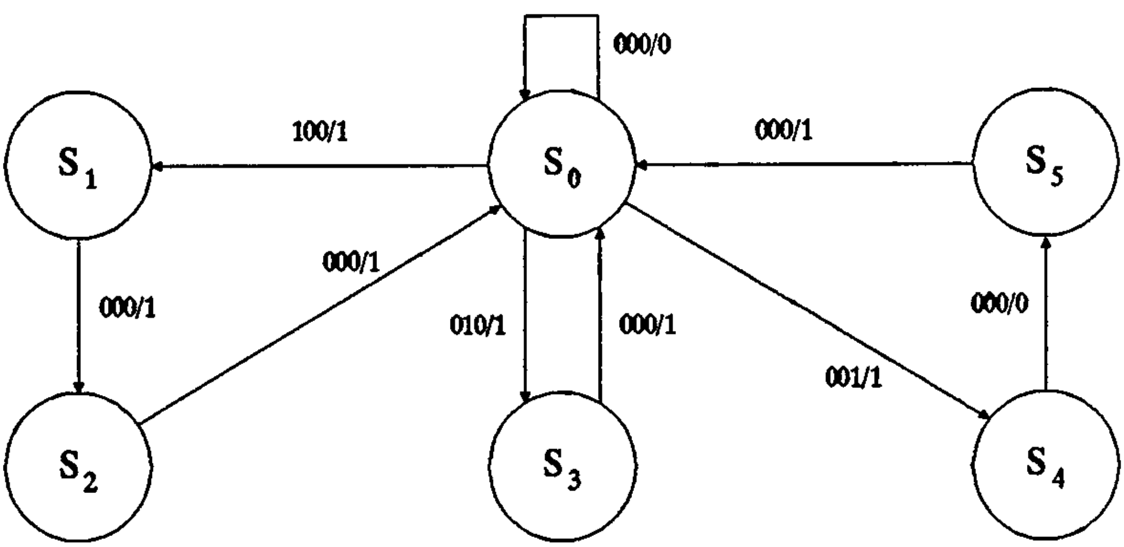
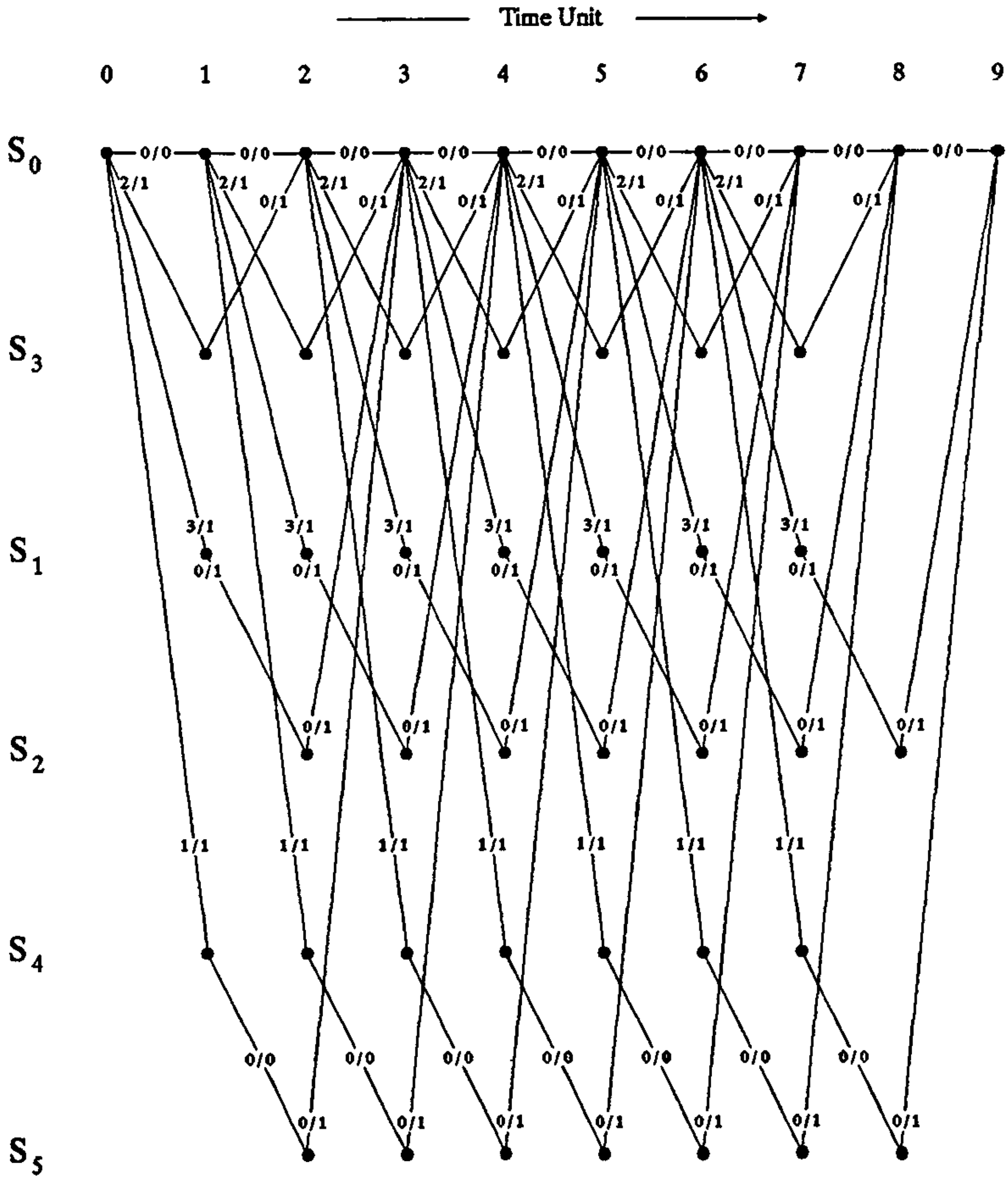


Figure A4.5.2: Constrained state-transition diagram (t=1) for the regulator circuit of Fig. A4.5.1.

The transition diagram of the constrained regulator circuit is shown in Fig. A4.5.2.

The corresponding trellis (see Fig. A4.5.3) follows readily from Fig. A4.5.2. Note that all transitions, except those originating from S_0, are caused by the all-zero I/P 3-tuple.



Notation for z_h: [000] = 0 [001] = 1 [010] = 2 [100] = 3

Figure A4.5.3: The trellis diagram corresponding to the constrained state-transition diagram of Fig. A4.5.2.

APPENDIX 4.6: EXAMPLE OF ERROR-TRELLIS SYNDROME DECODING

Example A4.6.1: Consider the (4,3,2) code of Example A4.5.1; its trellis diagram is shown in Fig. A4.5.3 (p. 360). Let the following source polynomial:

$$U(D) = \left[D+D^2+D^5+D^6, D+D^2+D^5+D^6, D+D^2+D^5+D^6\right] \qquad (A)$$

This is appended with mk zeros, to reset the encoder. Since this code is systematic, V(D) = [U(D),U(D)P(D)], where P(D) is given in Example A4.5.1 (p. 356). Then:

$$U(D)P(D) = \left[D+D^2+D^5+D^6, D+D^2+D^5+D^6, D+D^2+D^5+D^6\right]\begin{bmatrix} 1+D+D^2 \\ 1+D \\ 1+D^2 \end{bmatrix} \quad \Longleftrightarrow$$

$$U(D)P(D) = \left[D+D^2+D^5+D^6\right]\left[(1+D+D^2)+(1+D)+(1+D^2)\right] \quad \Longleftrightarrow$$

$$U(D)P(D) = \left[D+D^2+D^5+D^6\right]\left[1\right] = \left[D+D^2+D^5+D^6\right] \quad \Longrightarrow$$

$$V(D) = \left[D+D^2+D^5+D^6, D+D^2+D^5+D^6, D+D^2+D^5+D^6, D+D^2+D^5+D^6\right] \qquad (B)$$

This code is one-error correcting, hence it can correct a single error anywhere within a 12-bit sequence [= actual constraint-length, $n_A \hat{=} n(m+1)$]. Consider two channel errors, say, in the 3rd bit of the 3rd block and the 1st bit of the 7th block. Then, the error polynomial is:

$$E(D) = \left[D^6, 0, D^2, 0\right] \qquad (C)$$

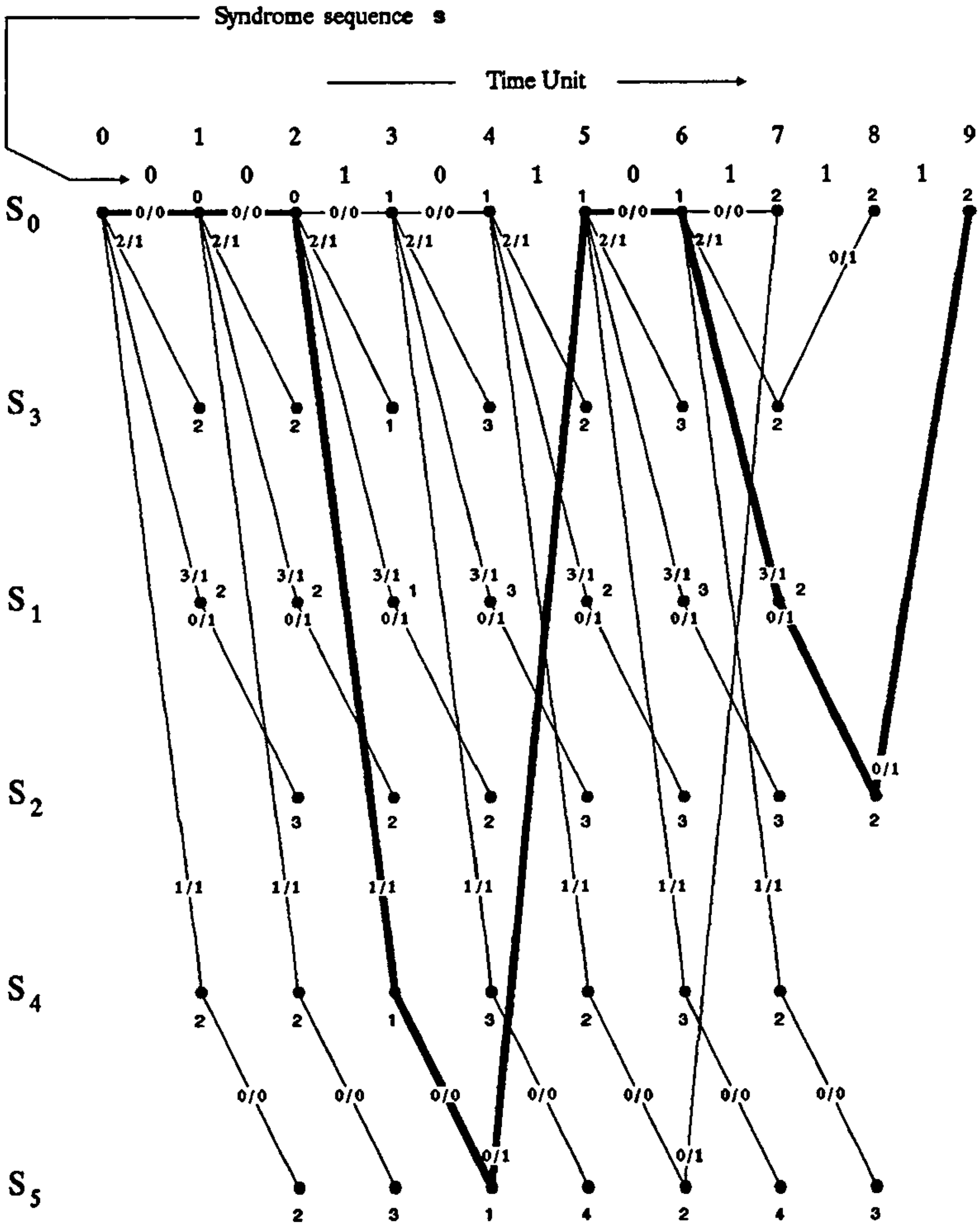
According to the decoding algorithm (see Note 4.6, p. 91), the decoder needs the syndrome S(D). Since the code is systematic, according to Lemma 2.11 (p. 48), $S(D) = E^{(p)}(D) - E^{(m)}(D)P(D)$. Then:

$$S(D) = \left[0\right] - \left[D^6, 0, D^2\right]\begin{bmatrix} 1+D+D^2 \\ 1+D \\ 1+D^2 \end{bmatrix} \quad \Longleftrightarrow$$

$$S(D) = D^6(1+D+D^2) + D^2(1+D^2) = D^2 + D^4 + D^6 + D^7 + D^8 \qquad (D)$$

S(D), from (D), corresponds to the syndrome sequence (note that since n-k = 1, the syndrome sequence is organized in one-bit blocks):

$$\mathbf{s} = \left(0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 0\ 0\ \cdots\right) \tag{E}$$



Notation for z_h: [000] = 0 [001] = 1 [010] = 2 [100] = 3

Figure A4.6.1: Error-trellis syndrome decoding of a 36-bit channel sequence corrupted by 2 errors (for the code with the trellis of Fig. A4.5.3).

Using (E) with the trellis of Fig. A4.5.3, the hyperchannel error sequence may be obtained (see Fig. A4.6.1).

The final survivor is the path with z-label 0-0-1-0-0-0-3-0-0 (highlighted).

The best estimate of the hyperchannel error sequence is found to be $\tilde{z}$ = (000 000 001 000 000 000 100) (the last six zeros are not part of the message sequence), which corresponds to the hyperchannel error polynomial:

$$\tilde{Z}(D) = \left[D^6, 0, D^2\right] \qquad (F)$$

Comparing (F) with (C), it becomes obvious that the decoder located both errors.

APPENDIX 4.7: PROOF OF THE THEORY IN PARAGRAPH 4.4.1.

A4.7.1. Proof of Theorem 4.12

From eqns (3.5) (p. 58):

$$FEG \cup CEG \cup REG = \left\{u_{h-1}^{(i)}, u_{h-j}^{(i)}, u_{h-M_i}^{(i)} \quad /i \text{ satisfies } X,\ j\in[2,M_i)\right\}$$

$$\Longleftrightarrow$$

$$FEG \cup CEG \cup REG = \left\{u_{h-j}^{(i)} \quad /i \text{ satisfies } X,\ j\in[1,M_i]\right\} \qquad (A)$$

where condition X is: $i\in[1,k]$ & $M_i\geq 1$ OR
$i\in[1,k]$ & $M_i\geq 1$ OR
$i\in[1,k]$ & $M_i\geq 3$

Condition X above is obviously equivalent to condition $i\in[1,k]$ & $M_i\geq 1$, hence eqn (A) above gives:

$$FEG \cup CEG \cup REG = \left\{u_{h-j}^{(i)} \ /i=1,2,..,k : M_i\geq 1,\ j=1,2,..,M_i\right\} = MEG$$

by eqn (3.5a). From eqns (3.5b) & (3.5d):

$$\left.\begin{array}{l} FEG = \left\{u_{h-j}^{(i)} \quad /i\in[1,k] \ \& \ M_i\geq 1,\ j\in[1,1]\right\} \\ CEG = \left\{u_{h-j}^{(i)} \quad /i\in[1,k] \ \& \ M_i\geq 3,\ j\in[2,M_i)\right\} \end{array}\right] \rightarrow \quad \Longrightarrow$$

$$\Longrightarrow \quad FEG \cap CEG = \left\{u_{h-j}^{(i)} \ /i\in[1,k] \ \& \ M_i\geq 3,\ j=1 \ \& \ 2\leq j\leq M_i-1\right\} = \emptyset$$

From eqns (3.5c) & (3.5d):

$$\left.\begin{array}{l} REG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 1,\ j\in[M_i, M_i]\right\} \\ CEG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 3,\ j\in[2, M_i)\right\} \end{array}\right] \longrightarrow \quad \longrightarrow$$

$$\longrightarrow \quad REG \cap CEG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 3,\ j=M_i\ \&\ 2\leq j\leq M_i-1\right\} = \emptyset$$

From eqns (3.5b) & (3.5c):

$$\left.\begin{array}{l} FEG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 1,\ j=1\right\} \\ REG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 1,\ j=M_i\right\} \end{array}\right] \longrightarrow \quad \longrightarrow$$

$$\longrightarrow \quad FEG \cap REG = \left\{u^{(i)}_{h-j} \quad / i\in[1,k]\ \&\ M_i\geq 1,\ j=1=M_i\right\} \quad \longrightarrow$$

$$FEG \cap REG = \left\{u^{(i)}_{h-1} \quad / i\in[1,k]\ \&\ M_i=1\right\}$$

QED

A4.7.2. Proof of Theorem 4.13

Consider the set X = FEG ∪ CEG ∪ REG' and use eqn (A4.2.11) and the distributive law [eqn (A4.2.5b)].

$$\begin{aligned} X &= FEG \cup CEG \cup (REG - FEG) = FEG \cup CEG \cup (REG \cap -FEG) \\ &= [(FEG \cup CEG) \cup REG] \cap [(FEG \cup CEG) \cup (-FEG)] \\ &= (FEG \cup CEG \cup REG) \cap [(CEG \cup (FEG \cup -FEG)] \qquad (A) \end{aligned}$$

by the associative (A4.2.4a) and commutative (A4.2.3) laws.

In eqn (A), the 1st parenthesis gives MEG (by Theorem 4.12), while the 2nd, CEG ∪ MEG [by eqn (A4.2.8)].

$$X = MEG \cap (CEG \cup MEG) = MEG \cap MEG = MEG \quad \longrightarrow$$

$$\longrightarrow \quad FEG \cup CEG \cup REG' = MEG \qquad (B)$$

Consider now the intersections of CEG, REG' & FEG:

From eqn (4.31b): $FEG \cap CEG = \emptyset$ (C)

$$\begin{aligned} FEG \cap REG' &= FEG \cap (REG - FEG) = FEG \cap (REG \cap -FEG) \\ &= FEG \cap (-FEG \cap REG) \quad \text{[by eqn (A4.2.3)]} \\ &= (FEG \cap -FEG) \cap REG \quad \text{[by eqn (A4.2.4b)]} \\ &= \emptyset \cap REG = \emptyset \quad \text{[by eqn (A4.2.7)]} \quad \longrightarrow \end{aligned}$$

$$\longrightarrow \quad FEG \cap REG' = \emptyset \qquad (D)$$

$$CEG \cap REG' = CEG \cap (REG \cap -FEG)$$

$= (CEG \cap REG) \cap -FEG$ [by eqn (A4.2.4b)]

$= \emptyset \cap -FEG = \emptyset$ [by eqns (4.31b) & (A4.2.7)]

$$\Longrightarrow \quad CEG \cap REG' = \emptyset \qquad (E)$$

It is evident from eqns (B), (C), (D) & (E) and Definition A4.2.3 that MEG = <FEG,CEG,REG'>. Reln (b) is proved similarly.

QED

A4.7.3. Proof of Theorem 4.14

Because FEG' is a subset of FEG and FEG & CEG are parts of the encoder memory, while MIG is not,

$$MIG(h) \cap FEG'(h) = \emptyset \qquad (A)$$

$$MIG(h) \cap CEG(h) = \emptyset \qquad (B)$$

Also, by Definition 3.1 (p. 58), CEG "···contains the stages that do not belong to either the FEG or the REG.". Hence, since FEG' is a subset of FEG,

$$FEG'(h) \cap CEG(h) = \emptyset \qquad (C)$$

What remains to be done is to prove that the union of the three mutually exclusive sets is MEG(h+1).

From eqns (4.33) & (A4.2.11),

$$FEG' = FEG - REG = FEG \cap -REG$$

and using Definition 3.1 (p. 58),

$$FEG' = \left\{u_{h-j}^{(i)} \;/ i=1,2,\ldots,k : M_i \geq 1,\ j=1 \ \&\ j<M_i\right\} \Longrightarrow$$

$$FEG' = \left\{u_{h-1}^{(i)} \;/ i=1,2,\ldots,k : M_i \geq 2\right\} \qquad (4.35)$$

From eqns (4.23b), (4.35) & (3.5d):

$$MIG(h) \cup FEG'(h) \cup CEG(h) = \left\{u_{h-j}^{(i)} \;/ i\in[1,k],\ (j=0 \ \&\ M_i \geq 1) \text{ OR } (j=1 \ \&\ M_i \geq 2) \text{ OR } (j\in[2,M_i) \ \&\ M_i \geq 3)\right\} \Longrightarrow$$

$$MIG(h) \cup FEG'(h) \cup CEG(h) = \left\{u_{h-j}^{(i)} \;/ i\in[1,k],\ j\in[0,M_i) \ \&\ M_i \geq 1\right\} \qquad (4.36)$$

From eqn (3.5a) (see p. 58):

$$MEG(h+1) = \left\{u^{(i)}_{h+1-j} \quad / i\in[1,k],\ M_i\geq 1\ \&\ j\in[1,M_i]\right\} \qquad (\text{let } j-1=\alpha)$$

$$MEG(h+1) = \left\{u^{(i)}_{h-\alpha} \quad / i\in[1,k],\ M_i\geq 1\ \&\ \alpha\in[0,M_i)\right\}$$

From the last expression and eqn (4.36),

$$MIG(h) \cup FEG'(h) \cup CEG(h) = MEG(h+1)$$

QED

A4.7.4. Proof of Lemma 4.10

From Theorem 4.14,

$$\begin{aligned} MEG(h+1) \cup DIG(h) &= MIG(h) \cup FEG'(h) \cup CEG(h) \cup DIG(h) \\ &= [MIG(h) \cup DIG(h)] \cup FEG'(h) \cup CEG(h) \\ &= ING(h) \cup FEG'(h) \cup CEG(h) \end{aligned}$$

Note also that, FEG'(h) & CEG(h) are mutually exclusive, since they partition a set (see Theorem 4.14), and that ING $\cap$ FEG' = ING $\cap$ CEG = $\emptyset$, because ING does not belong to the circuit memory, of which FEG' & CEG are parts.

QED

APPENDIX 4.8: PROOF OF THEOREM 4.15

It is known, from Lemma 4.9 (p. 89), that the total number of different states that can be reached within one time-unit from a state of weight w, is denoted by $\Sigma¥(w)$ and an expression is given by eqn (4.30c). The task therefore is to prove that the total number of states from which a state of weight w can be reached is also $\Sigma¥(w)$.

From Lemma 4.10,

$$MEG(h+1) \cup DIG(h) = \langle ING(h), FEG'(h), CEG(h)\rangle \qquad (A)$$

Since DIG(h) contains bits that are not stored in the memory, then DIG(h) $\cap$ MEG(h+1) = $\emptyset$ hence, from eqn (A),

$$\langle MEG(h+1), DIG(h)\rangle = \langle ING(h), FEG'(h), CEG(h)\rangle \qquad (B)$$

Applying Theorem 4.9 (p. 87) into eqn (B),

$$w\left[S(h+1)\right] + w\left[D(h)\right] = w\left[z_h\right] + w\left[F'(h)\right] + w\left[C(h)\right] \qquad (C)$$

From Theorem 4.10 (p. 88),

$$w\big[S(h)\big] + w\big[z_h\big] \le t \qquad (D)$$

Since w[S(h+1)] = w, from (C) & (D),

$$w\big[S(h)\big] + w + w\big[D(h)\big] - w\big[F'(h)\big] - w\big[C(h)\big] \;\le\; t \qquad (E)$$

Consider Partition II of MEG(h) (see Theorem 4.13, p. 93) and apply Theorem 4.9 (p. 87):

$$MEG(h) = \langle FEG'(h), CEG(h), REG(h) \rangle \implies$$

$$w\big[S(h)\big] - w\big[F'(h)\big] - w\big[C(h)\big] = w\big[R(h)\big] \qquad (F)$$

From (E) & (F), since $w[\cdots] \ge 0$:

$$w\big[R(h)\big] \le t - w\big[S(h+1)\big] - w\big[D(h)\big] \le t - w\big[S(h+1)\big] \qquad (4.38)$$

Note from Theorem 4.14 (p. 95) that, the bits that make up the state at time-unit h+1 are those belonging to MIG(h), FEG'(h) & CEG(h). Since FEG'(h), CEG(h) & REG(h) partition MEG(h) then the only memory bits, of the current state, that do not participate in the formation of the next state are the REG(h) ones, and only those. Hence, the states from which one can reach a specific next state, $S(h+1) = S_y$, should equal the total number of different R(h)s. Note that the Hamming weight of R(h) is bounded by (4.38).

If w[R(h)]=i, since REG has k-ƒ elements, there are C(k-ƒ,i) different R(h)s of weight i, and since there are

$$\sum_{i=0}^{t-w} C(k-f,i)$$

different R(h)s in all, there are as many states from which $S(h+1) = S_y$, a state of weight w, can be reached.

Note that the above analysis is valid only <u>within</u> the central portion of the trellis, i.e. not for time-units $\le$ m or $\ge$ L. This is so because in calculating the number of states from which any particular state S(h) can be reached, one considers a transition $S(h-1) \longrightarrow S(h)$, where $h-1 \ge m \implies h > m$. Also, in calculating the number of states that can be reached from any particular state S(h), one considers transitions of the type $S(h) \longrightarrow S(h+1)$, where $h+1 \le L \implies$

h < L.

QED

APPENDIX 4.9: THE INTERMEDIATE RESULTS OF § 4.6.3.

A4.9.1. Proof of Theorem 4.25

Relns (a) & (b) follow easily from Definition 4.8. Reln (c) is based on Theorem 4.19:

$$\sum_{i=1}^{m} i f(i) = M$$

Substituting $f(i) = F(i) - F(i-1)$ [(4.55a)],

$$\sum_{i=1}^{m} iF(i) - \sum_{i=1}^{m} iF(i-1) = M \iff$$

$$\sum_{i=1}^{m} iF(i) - \sum_{j=1}^{m-1} jF(j) - \sum_{j=0}^{m-1} F(j) = M \iff$$

$$mF(m) - \sum_{j=0}^{m} F(j) + F(m) = M$$

Since $F(m) = k$ [reln (b)]

$$\sum_{j=0}^{m} F(j) = (m+1)k - M$$

QED

A4.9.2. Proof of Lemma 4.13

Let $A(\beta) \triangleq \sum_{i=1}^{\beta} i f(i)$ and use eqn (4.55a):

$$A(\beta) = \sum_{i=1}^{\beta} iF(i) - \sum_{i=1}^{\beta} iF(i-1) \iff$$

$$A(\beta) = \sum_{i=1}^{\beta} iF(i) - \sum_{j=1}^{\beta-1} jF(j) - \sum_{j=0}^{\beta-1} F(j) = \beta F(\beta) - \sum_{j=0}^{\beta-1} F(j)$$

Relation (b) is easily proved using the above result and reln (4.42e):

$$\sum_{i=\beta}^{m} if(i) = \sum_{i=1}^{m} if(i) - A(\beta-1) = M - A(\beta-1)$$

QED

A4.9.3. Proof of Lemma 4.14

For any two sets X & Y, it is accepted that:

$$\text{if } X \cap Y = \emptyset, \text{ then } |X| + |Y| = |X \cup Y| \qquad (4.58)$$

For a proof of eqn (4.58) (which is trivial anyway) see Biggs [36], p. 44.

From Theorem A4.2.1, A U B is partitioned into A-B & B.

So, $(A-B) \cap B = \emptyset$ (A)

and $(A-B) \cup B = A \cup B$ (B)

From eqn (B): $|A \cup B| = |(A-B) \cup B|$ and using eqn (A) in (4.58): $|(A-B) \cup B| = |A - B| + |B|$, from which eqn (a) follows. Eqn (b) is a special case of (A), because if $B \subseteq A$, then $A \cup B = A$.*

QED

APPENDIX 4.10: CONSTRAINED & SIMPLIFIED STATE-TRANSITION DIAGRAMS FOR A t=2 NORMAL LSC

Example A4.10.1: Consider the state-transition diagram of Fig. A3.1.2 (p. 332). It corresponds to a (4,3,2) normal LSC, with total memory M=3, shown in Fig. A2.9.2 (p. 315).

Let a weight-constraint of 2 be imposed on it. Then, in its transition diagram, the sum of the Hamming weight of the current state plus the current input-block should not exceed 2 (see Theorem 4.10, p. 88).

Hence, state S_7 is removed. For the remaining states, only those transitions (out of each state) satisfying $w[S(h)] + w[z_h] \le 2$ are kept. Hence, from the weight-2 states (S_3,S_5 & S_6) only one transition is kept (corresponding to z_h = [000]). From the weight-1 states (S_1,S_2 & S_4),

* Remember, $A \subseteq B$ denotes "A is a *subset* of B".

only transitions of weight 0 or 1 are kept. From S_0, transitions of weight 0, 1, or 2 are kept (see Fig. A4.10.1).

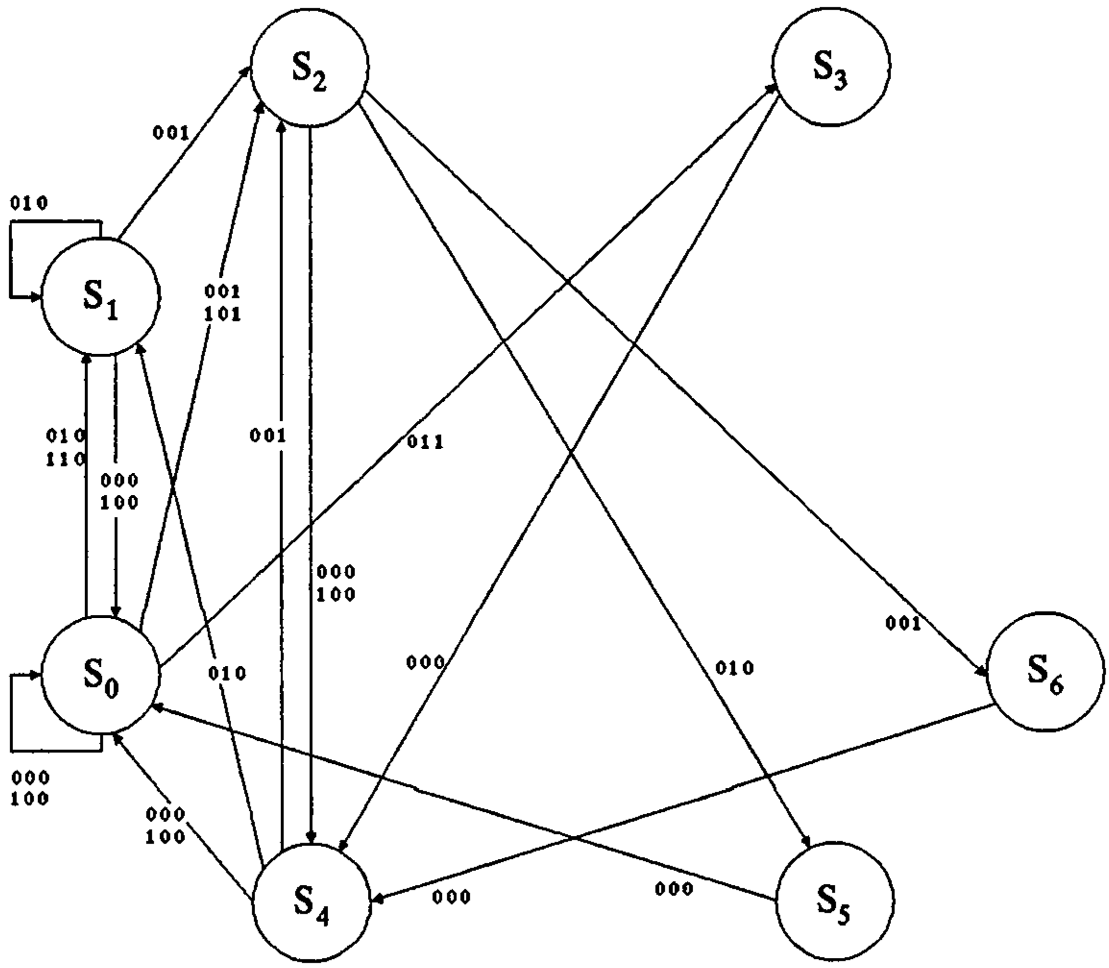


Figure A4.10.1: t=2 constrained state-transition diagram, for the (4,3,2) normal LSC of Fig. A2.9.2 (p. 315).

Consider now the application of Theorem 4.29, on the above diagram. From the circuit diagram:

S = [CBA]   R(h) = [CA] f = 1   k = 3   &   M = 3

There is one state of weight 0, S_0,
with one transition to itself [σ = 1+C(f,1)+C(f,2) = 1+C(1,1)+C(1,2) = 2],
k-f=2 transitions to the weight-1 region [σ=1+min{1,f}=2] (to S_1 & S_2) and
if k-f=2≥2, (k-f)(k-f-1)/2=1 transition to the weight-2 region [σ=1] (to S_3).

There are M=3 states of weight 1 (S_1,S_2 & S_4).

$M-k+f=1$ of them has one transition to a weight-1 state $[\sigma = 1+\min\{1,f\} = 2]$:

$S_2 \longrightarrow S_4$; $S_2=[CBA]=[010] \Longrightarrow R(h)=[CA]=[00] \Longrightarrow \tau=0$ $(\sigma=2)$

and $k-f=2$ transitions to the weight-2 region $[\sigma=1]$ (S_5 & S_6).

The rest, $k-f=2$, states have one transition, each, to S_0 $[\sigma = 1+\min\{2,f\} = 2]$:

$S_1 \longrightarrow S_0$; $S_1=[CBA]=[001] \Longrightarrow R(h)=[CA]=[01] \Longrightarrow \tau=1$ $(\sigma=2)$

$S_4 \longrightarrow S_0$; $S_4=[CBA]=[100] \Longrightarrow R(h)=[CA]=[10] \Longrightarrow \tau=1$ $(\sigma=2)$

and $k-f=2$ transitions, each, to a weight-1 state $[\sigma=1]$:

$S_1 \longrightarrow S_1$; $S_1=[CBA]=[001] \Longrightarrow R(h)=[CA]=[01] \Longrightarrow \tau=1$ $(\sigma=1)$

$S_1 \longrightarrow S_2$; $S_1=[CBA]=[001] \Longrightarrow R(h)=[CA]=[01] \Longrightarrow \tau=1$ $(\sigma=1)$

$S_4 \longrightarrow S_1$; $S_4=[CBA]=[100] \Longrightarrow R(h)=[CA]=[10] \Longrightarrow \tau=1$ $(\sigma=1)$

$S_4 \longrightarrow S_2$; $S_4=[CBA]=[100] \Longrightarrow R(h)=[CA]=[10] \Longrightarrow \tau=1$ $(\sigma=1)$

There are $M(M-1)/2=3$ states of weight 2 (S_3,S_5,S_6), with one transition each $[\sigma=1]$:

Provided that $M-2 \geq k-f \Longrightarrow 1 \geq 2$, $(M-k+f)(M-k+f-1)/2$ states transit to another weight-2 state.

Provided that $M-1 \geq k-f \Longrightarrow 2 \geq 2$, $(k-f)(M-k+f)=2$ states transit to a weight-1 state:

$S_3 \longrightarrow S_4$; $S_3=[CBA]=[011] \Longrightarrow R(h)=[CA]=[01] \Longrightarrow \tau=1$ $(\sigma=1)$

$S_6 \longrightarrow S_4$; $S_6=[CBA]=[110] \Longrightarrow R(h)=[CA]=[10] \Longrightarrow \tau=1$ $(\sigma=1)$

Provided that $k-f=2 \geq 2$, the remaining, $(k-f)(k-f-1)/2=1$, state transits to S_0:

$S_5 \longrightarrow S_0$; $S_5=[CBA]=[101] \Longrightarrow R(h)=[CA]=[11] \Longrightarrow \tau=2$ $(\sigma=1)$

Hence, the results of Theorem 4.29 were verified via the above example. If the total number of transitions/diagram is considered as a complexity measure [= (transitions/state) x (No of states)], then:

The unconstrained transition diagram has (see Fig. A3.1.2, p. 332) $2 \times 4 \times 8 = 64$ transitions.

The $t=2$ constrained state-transition diagram has (counting state-by-state, starting from $S_0,S_1,\ldots,S_6$ - see Fig. A4.10.1), $(2+2+2+1) + (2+1+1) + (1+1+1) + (1) + (2+1+1) + (1) + (1) = 21$ transitions, or about 1/3 of the unconstrained.

▮

Example A4.10.2: Consider the constrained state-transition diagram of Fig. A4.10.1. The simplified diagram is easily constructed following the instructions of Note 4.7 (p. 99). The result is shown in Fig. A4.10.2.

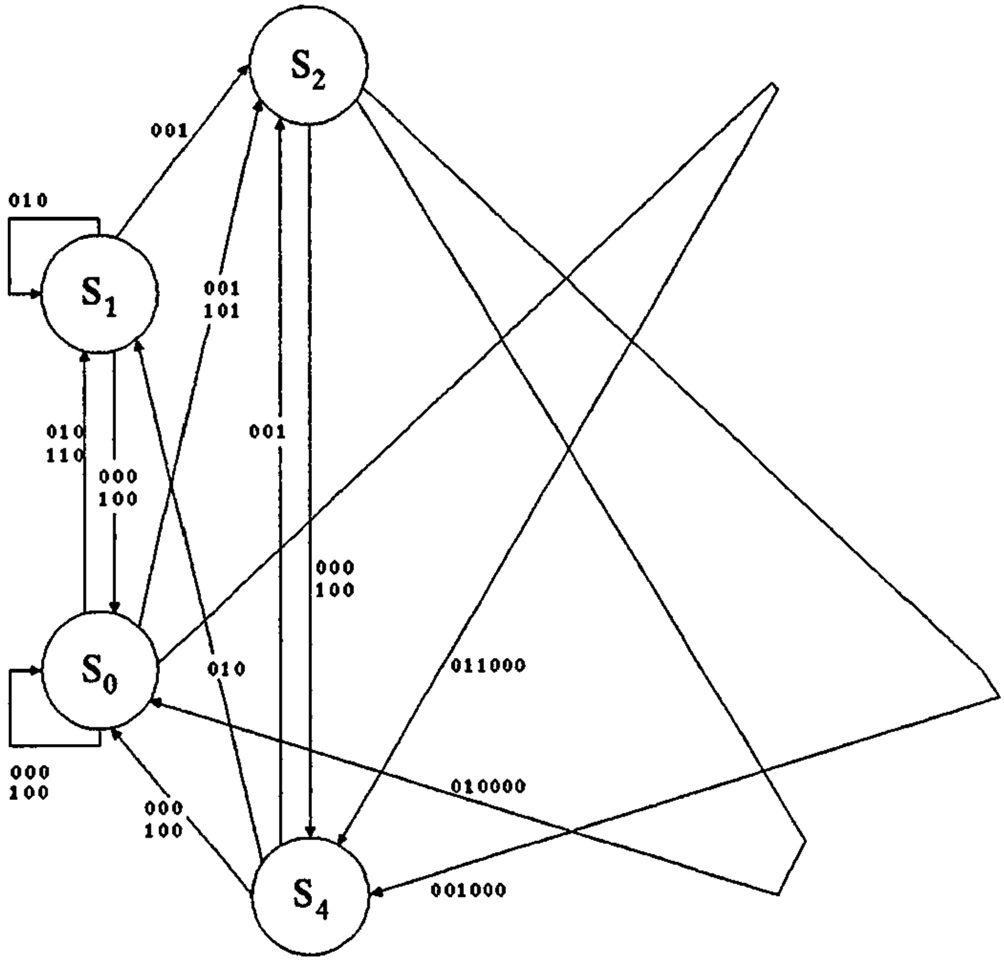


Figure A4.10.2: t=2 simplified state-transition diagram, for the (4,3,2) normal LSC of Fig. A2.9.2 (p. 315).

Since the circuit is a (4,3,2) one, according to Theorem 4.18, its longest transition is m+1, only if the LSC contains at least t=2 SRs of length m. Hence, the simplified state-transition diagram will not contain transitions of length 3, as can be verified from Fig. A4.10.2. Hence, the only long transitions are the β=2 ones.

Consider the application of Theorem 4.27. The states to be examined may have weight w∈[0,t], i.e. w = 0,1,2.

The memory details are (from Fig. A2.9.2, p. 315):

$$F(0) = F(1) = F(2) = 1 \quad M = 3 \quad \& \quad k = 3$$

i) The number of states of weight w, from which a transition of length $\beta = 2$ may start is

$$\binom{|{-}REG(2-1,)|}{w} \tag{4.60a}$$

where: $$|{-}REG(2-1,)| = 3 - (2-1)3 + \sum_{i=0}^{2-2} F(i) \tag{4.60b}$$

$|{-}REG(1,)| = 3 - 3 + F(0) = 1$. Hence w = 0 or 1. Then:

There is $\binom{1}{0} = 1$ state (S_0), of weight 0, from which β=2 transitions may start (to S_4).

There is $\binom{1}{1} = 1$ state (S_2), of weight 1, from which β=2 transitions may start (to S_0 & S_4).

ii) There are $$\binom{|{-}REG(2-1,)|}{w} - \binom{|{-}REG(2,)|}{w} \tag{4.60c}$$

states, of weight w, from which

$$\binom{|{-}DIG(2-2,)|}{2-w} = \binom{3-F(2-2)}{2-w} \tag{4.60d}$$

transitions of length β=2 start, where:

$$|{-}REG(2,)| = |{-}REG(2-1,)| + F(2-1) - 3 \tag{4.60e}$$

$|{-}REG(2,)| = 1 + 1 - 3 = -1$. Hence the 2nd term of (4.60c) drops, and since $|{-}REG(1,)| = 1$:

There is $\binom{|{-}REG(2-1,)|}{w} = 1$ state of weight w (=0,1), from which

$$\binom{|{-}DIG(2-2,)|}{2-w} = \binom{3-F(2-2)}{2-w} = \binom{2}{2-w} = 2/[(2-w)w!)] = 2/(2-w)$$

β=2 transitions start:

One state of weight 0 (S_0) from which 2/(2-0) = 1, β=2 transition starts (to S_4).

One state of weight 1 (S_2) from which 2/(2-1) = 2, β=2 transitions start (to S_0 & S_4).

iii) There are $$\binom{|{-}REG(2,)|}{w} \tag{4.60f}$$

states, of weight w, from which

$$\binom{|{-}DIG(2-2,)|}{2-w} - \binom{|{-}DIG(2-1,)|}{2-w}$$

$$\binom{3-F(2-2)}{2-w} - \binom{3-F(2-1)}{2-w} \tag{4.60d}$$

transitions of length $\beta = 2$ start.

Since $|\text{-REG}(2,)| = -1$, there is no β=2 transition, under this category.

Hence, Theorem 4.27 was verified by the above example.

∎

APPENDIX 4.11: PROOF OF THEOREMS 4.30 & 4.31

A4.11.1. Proof of Theorem 4.30

Since all SRs have the same length, this length is the maximum, i.e. m. Then $f = 0$ and $M = km$. Substitution of these values to the results of Theorem 4.21 (p. 107), gives the constrained trellis results.

Since all SRs have the same length, $f = f(0) = f(1) = \cdots = f(m-1) = 0$ and $f(m) = k$. Also, $F(0) = F(1) = \cdots = F(m-1) = 0$ and $F(m) = k$.

From Theorem 4.18 (p. 101), $\beta\in[2,m+1]$ and $\beta = m+1$, only if there exist at least t SRs of length m, which in the present case is equivalent to $t\leq k$.

From (4.60b) & (4.60e), for $\beta\in[2,m+1]$:

$$|\text{-REG}(\beta-1,)| = km - k(\beta-1) + 0 = k(m-\beta+1) \qquad \text{(A)}$$

$$|\text{-REG}(\beta,)| = k(m-\beta+1) + F(\beta-1) - k = k(m-\beta) + F(\beta-1) \qquad \text{(B)}$$

From (4.60d):

$$|\text{-DIG}(\beta-2,)| = k - F(\beta-2) = k \quad /\beta\in[2,m+1] \qquad \text{(C)}$$

and $$|\text{-DIG}(\beta-1,)| = k - F(\beta-1) \quad /\beta\in[2,m+1] \qquad \text{(D)}$$

From the above, for $\beta = m+1$:

$$|\text{-REG}(m,)| = |\text{-REG}(m+1,)| = 0 \qquad \text{(E)}$$

$$|\text{-DIG}(m-1,)| = k \quad \text{and} \quad |\text{-DIG}(m,)| = k - F(m) = 0 \qquad \text{(F)}$$

From (E) & (F), above, the 2nd part of Theorem 4.27 drops, because the number of states under this category is:

$$\binom{|\text{-REG}(m,)|}{w} - \binom{|\text{-REG}(m+1,)|}{w} = \binom{0}{w} - \binom{0}{w} = 0$$

Similarly, the number of states under the 3rd category is:

$\binom{|-REG(m+1,)|}{w} = \binom{0}{w} = 1$ if w=0, and = 0 if w>1. Then, there is only one state, S_0, from which transitions of length β=m+1 start. There are

$$\binom{|-DIG(m-1,)|}{t-w} - \binom{|-DIG(m,)|}{t-w} = \binom{k}{t} - \binom{0}{t} = \binom{k}{t}$$

transitions from S_0. This proves part (iii) of the the theorem.

If $\beta\in[2,m]$, from (C) & (D), $|-DIG(\beta-2,)| = |-DIG(\beta-1,)| = k$, since $F(\beta-1) = 0$. Hence there are no transitions from the states of the 3rd category (see Theorem 4.27). From the 2nd category, using (A), (B) & (C), there are

$$\binom{k(m-\beta+1)}{w} - \binom{k(m-\beta)}{w} \quad \text{states with} \quad \binom{k}{t-w}$$

transitions each. This proves part (iv) of the theorem. For the last part, let β = m, above. For w=0, there are C(k,0) - C(0,0) = 1 - 1 = 0, hence state S_0 does not have transitions of length m. For w>1, there are C(k,w) - C(0,w) = C(k,w) states, each with C(k,t-w) transitions.

QED

A4.11.2. Proof of Theorem 4.31

Theorem 4.31 is an application of Theorem 4.30, for the special case of k = 1. Part (i) is straightforward.

For part (ii): For each $\tau \in \left[\max\{0,\varsigma-m+1\}\ ,\ \min\{1,\varsigma\}\right]$,

there are $\binom{1}{\tau}\binom{m-1}{\varsigma-\tau}$ states of weight ς, each of which has

$\binom{1}{\tilde{n}+\tau-\varsigma}$ single-edge transitions, to the weight-$\tilde{n}$ region,

where: $\varsigma-\tau \le w\left[S(h+1)\right] \triangleq \tilde{n} \le MIN\{t-\tau,1+\varsigma-\tau\}$ $\Longleftrightarrow$

$0 \le \tilde{n}+\tau-\varsigma \le MIN\{t-\varsigma,1\}$.

Since τ = 0 or 1, $C(1,\tau)$=1. Since, also, $\tilde{n}+\tau-\varsigma$ = 0 or 1,

$C(1,\tilde{n}+\tau-\varsigma)$=1. Hence: For each $\tau \in \left[\max\{0,\varsigma-m+1\}\ ,\ \min\{1,\varsigma\}\right]$,

there are $\binom{m-1}{\varsigma-\tau}$ states of weight ς, each of which has one single-edge transition, to the weight-$\tilde{n}$ region, where: $\varsigma \le \tilde{n} + \tau \le \mathrm{MIN}\{t,\varsigma+1\}$.

If $\varsigma=0$, then $\tau\in[\max\{0,-m+1\},\min\{1,0\}] \Longrightarrow \tau=0$. There is $C(m-1,\varsigma-\tau) = C(m-1,0) = 1$ state of weight 0, with one single-edge transition to the weight-$\tilde{n}$ region, where: $\varsigma \le \tilde{n}+\tau \le \mathrm{MIN}\{t,\varsigma+1\} \iff 0 \le \tilde{n} \le \mathrm{MIN}\{t,1\} \iff \tilde{n} =$ 0 or 1. If $\tilde{n}=0$, the next state is S_0, while if $\tilde{n}=1$, the next state will be $[00\cdots01] = S_1$.

For the case $\varsigma<m$, if $\varsigma\in[1,t]$, for each $\tau \in [\max\{0,\varsigma-m+1\}, \min\{1,\varsigma\}] \iff \tau \in [0,1]$ (because, $\varsigma<m \Longrightarrow \varsigma-m+1<1$) there are $C(m-1,\varsigma-\tau)$ (which is non-zero because $\tau \ge \varsigma-m+1 \Longrightarrow m-1 \ge \varsigma-\tau$) states with one single-edge transition, each, to the weight-$\tilde{n}$ region, where: $\varsigma \le \tilde{n}+\tau \le \mathrm{MIN}\{t,\varsigma+1\}$. With respect to the last inequality, there are three possibilities: Either $t\ge\varsigma+1$, or $t=\varsigma$, or $t<\varsigma$. Since $\varsigma\in[1,t]$, either $t\ge\varsigma+1$, or $t=\varsigma$. Hence, if $\varsigma\in[1,t)$ then $\varsigma-\tau \le \tilde{n} \le \varsigma+1-\tau$, and if $t=\varsigma$ then $\tilde{n} = t-\tau$.

If $\tau=0$, there are $C(m-1,\varsigma)$ states of weight $\varsigma\in[1,t)$ with two single-edge transitions, each, to states of weight ς & $\varsigma+1$ and $C(m-1,t)$ states of weight t, with one single-edge transition to a state of weight t.

If $\tau=1$, there are $C(m-1,\varsigma-1)$ states of weight $\varsigma\in[1,t)$ with two single-edge transitions, each, to states of weight ς & $\varsigma-1$ and $C(m-1,t-1)$ states of weight t, with one single-edge transition to a state of weight $t-1$.

For the case $\varsigma\ge m$ (assuming of course that $t\ge m$), only the $\varsigma=m$ case is meaningful. Then, for each τ /$\max\{0,\varsigma-m+1\} \le \tau \le \min\{1,\varsigma\} \iff \tau=1$, there is $C(m-1,m-1) = 1$ state with one single-edge transition to the weight-$\tilde{n}$ region, where: $\varsigma \le \tilde{n}+\tau \le \mathrm{MIN}\{t,\varsigma+1\} \iff m \le \tilde{n}+1 \le \mathrm{MIN}\{t,m+1\} \iff m-1 \le \tilde{n} \le \mathrm{MIN}\{t-1,m\}$. Hence, if $t=m$ there is one transition to a weight-$(m-1)$ state, while if $t\ge m+1$ there are two transitions, to states of weight $m-1$ and m.

With respect to the central portion of the simplified trellis:

iii) There is $\binom{1}{t} = 1$ maximum-length transition

(β = m+1), starting from state S_0. Since C(1,t) = 0 for t>1, such a long transition exists only if t=1.

iv) There are $\binom{m-\beta+1}{\varsigma} - \binom{m-\beta}{\varsigma}$ states of weight ς, with $\binom{1}{t-\varsigma} = 1$ (only if $1 \geq t-\varsigma \Longleftrightarrow \varsigma \geq t-1 \Longrightarrow \varsigma$ = t or t-1) transition of length β each, where $\beta \in [2,m]$. For any states to exist, $m-\beta+1 \geq \varsigma \Longleftrightarrow \beta \leq m+1-\varsigma$.

v) From above, if the current state has weight ς the longest transition is m+1-ς, and because the smallest ς is t-1, the longest transition is m+2-t, starting from a state of weight t-1. Hence, the I/P must be 1, as well, in order to bring the memory into a weight-t state and hence start a long transition. Finally, for the longest transition to take place, the SR must have t-1 1s in the first t-1 stages, and that corresponds to state S_α, where $\alpha=2^{t-1}-1$.

QED

APPENDIX 5.1: PROOF OF THE THRESHOLD-DECODING THEOREMS

A5.1.1. Proof of Theorem 5.1

Assume that x of the error digits are non-zero, where $0 \le x \le \lfloor J/2 \rfloor$. Since either $e_m = 0$, or $e_m \neq 0$:

i) If $e_m = 0$, then x other error digits are non-zero. Hence no more than x composite parity-checks are affected (i.e. become non-zero). All the rest e_is are zero, hence the rest J-x $\AE_i$s are zero. Since $x \le \lfloor J/2 \rfloor \le J/2 \implies J-x \ge J/2 \implies x \le J-x$. So, the majority vote is 0, unless x = J-x in which case there is a tie which is resolved in favour of 0. Hence, decoding is correct if conditions are satisfied and $e_m = 0$.

ii) If $e_m = V \neq 0$, then x-1 other error digits are non-zero. These x-1 digits affect at most x-1 $\AE_i$s. The rest J-x+1 $\AE_i$s are affected only by e_m, hence their value is V. Now, since $x-1 < \lfloor J/2 \rfloor \le J/2 \implies J-x+1 > J/2 > x-1$, hence the majority of the composite parity checks, vote for V, so decoding is correct if conditions are satisfied and $e_m = V \neq 0$.

From (i) & (ii), above, the theorem follows.

QED

A5.1.2. Proof of Theorem 5.2

According to Definition 5.3, the APP rule maximizes the conditional probability $P(e_m=V|\{\AE_i\})$. Consider

$$\textit{Baye's rule}: \quad P(A|B) = P(B|A)P(A)/P(B) \qquad \text{(A5.1.1)}$$

$$\text{Then,} \quad P\left(e_m=V|\{\AE_i\}\right) = P\left(\{\AE_i\}|e_m=V\right)P(e_m=V)/P(\{\AE_i\}) \qquad \text{(A)}$$

Since the error digits are statistically independent and the composite parity-checks $\AE_i$ are all orthogonal on e_m,

$$P\left(e_m=V|\{\AE_i\}\right) = \prod_{i=1}^{J} P\left(\AE_i|e_m=V\right)P(e_m=V) \;/\; \left[\prod_{i=1}^{J} P(\AE_i)\right] \implies$$

$$\Longrightarrow \quad \log P\left(e_m=V \mid \{\mathcal{E}_i\}\right) = \log P(e_m=V) + \sum_{i=1}^{J}\log P\left(\mathcal{E}_i \mid e_m=V\right) - \sum_{i=1}^{J}\log P(\mathcal{E}_i) \qquad (B)$$

Note that the 3rd term in the RHS of eqn (B) does not depend on e_m, i.e. varying the value of this digit will have no effect on this term. Hence, maximizing the conditional probability $P(e_m|\{\mathcal{E}_i\})$ is equivalent to maximizing its log (since the latter is a continuously increasing function of its argument) which is equivalent to maximizing the RHS of (B); since also varying e_m has no effect on $P(\mathcal{E}_i)$, the last term of the RHS of (B) may be ignored.

QED

A5.1.3. Proof of Theorem 5.3

Assume that x of the error bits are 1, where $0 \le x \le \lfloor J/2 \rfloor$. Since, either $e_m = 0$ or $e_m = 1$:

i) If $e_m = 0$, then x other error bits are non-zero. Hence no more than x composite parity-checks are 1. All the rest e_is are zero, hence the rest J-x $\mathcal{E}_i$s are zero. So, $\Sigma \le x \le \lfloor J/2 \rfloor \le \lceil J/2 \rceil \Longrightarrow e_m = 0$.

ii) If $e_m = 1$, then all $\mathcal{E}_i$s have one bit equal to 1 (e_m) and at most x-1 of them also have another bit equal to 1, which cancels out the e_m=1. So, at most x-1 $\mathcal{E}_i$s are zero or, the same, at least J-x+1 are 1. Hence,

$$\Sigma \ge J-x+1 \qquad (A)$$

Since $x \le \lfloor J/2 \rfloor \Longrightarrow -x \ge -\lfloor J/2 \rfloor \Longrightarrow$

$$\Longrightarrow \quad J-x+1 \ge J-\lfloor J/2 \rfloor+1 > J-\lfloor J/2 \rfloor$$

and combining with reln (A),

$$\Sigma > J-\lfloor J/2 \rfloor \; \begin{cases} J-J/2 = J/2 = \lceil J/2 \rceil & /J=\text{even} \\ J-(J-1)/2 = (J+1)/2 = \lceil J/2 \rceil & /J=\text{odd} \end{cases} \Longrightarrow$$

$$\Longrightarrow \quad \Sigma > \lceil J/2 \rceil \quad \Longrightarrow \quad e_m = 1$$

Note that application of the decoding criterion, (5.4), coupled with a restriction on the number of errors, led to correct decoding.

QED

A5.1.4. Proof of Theorem 5.4

According to the definition of p_i, the probability that $Æ_i = 0$, given that $e_m = 1$, equals the probability of an odd number of 'ones' in the rest of the error bits that participate in the formation of $Æ_i$, i.e. equals p_i. Similarly,

$$P\left(Æ_i=0 \mid e_m=1\right) = P\left(Æ_i=1 \mid e_m=0\right) = p_i$$

$$P\left(Æ_i=1 \mid e_m=1\right) = P\left(Æ_i=0 \mid e_m=0\right) = q_i$$

From Theorem 5.2, because $e_m = 0$ or 1, the APP decoding rule becomes:

Choose $e_m = 1$, iff

$$\log P(e_m=1) + \sum_{i=1}^{J} \log P\left(Æ_i \mid e_m=1\right) > \log P(e_m=0) + \sum_{i=1}^{J} \log P\left(Æ_i \mid e_m=0\right)$$

$$\Longleftrightarrow \quad \sum_{i=1}^{J} \log\left[P\left(Æ_i \mid e_m=1\right)/P\left(Æ_i \mid e_m=0\right)\right] > \log(q_0/p_0) \qquad \text{(A)}$$

Consider now the ratio $P\left(Æ_i \mid e_m=1\right)/P\left(Æ_i \mid e_m=0\right)$.

If $Æ_i = 0$ then:

$$P\left(Æ_i \mid e_m=1\right) \,/\, P\left(Æ_i \mid e_m=0\right) = P\left(Æ_i=0 \mid e_m=1\right) \,/\, P\left(Æ_i=0 \mid e_m=0\right) = p_i/q_i$$

If $Æ_i = 1$ then

$$P\left(Æ_i \mid e_m=1\right) \,/\, P\left(Æ_i \mid e_m=0\right) = P\left(Æ_i=1 \mid e_m=1\right) \,/\, P\left(Æ_i=1 \mid e_m=0\right) = q_i/p_i$$

Then, one may write:

$$P\left(Æ_i \mid e_m=1\right) \,/\, P\left(Æ_i \mid e_m=0\right) = \left(q_i/p_i\right)^{2Æ_i-1} \qquad \text{(B)}$$

From results (A) & (B), the condition becomes

$$\text{(A)} \quad \Longleftrightarrow \quad \sum_{i=1}^{J}(2Æ_i-1)\log(q_i/p_i) > \log(q_0/p_0)$$

from which condition (5.4) follows.

QED

APPENDIX 5.2: DEFINITE DECODING - PARITY SQUARES

In this appendix, the general case of estimating a given error block e_a /$a \geq m$ will be discussed. From Theorem 2.15 (p. 50), with $\theta \triangleq MIN\{h,m\}$:

$$s_h^{(j)} = e_h^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{\theta} e_{h-z}^{(i)} g_{k+j,z}^{(i)} \qquad /1 \leq j \leq n-k \qquad (A5.2.1)$$

In order to consider all syndrome blocks that check on e_a, one must require $h-z = a$, hence h should vary between $MIN\{a+z\}$ and $MAX\{a+z\}$, i.e. from a to $a+\theta = a+MIN\{h,m\}$: $a \leq h \leq a+MIN\{h,m\}$ and since $h \geq a \geq m$, $\theta=m$ and h should be allowed to range from a to $a+m$. Hence, the set of syndrome bits that may check on error bit $e_a^{(\mu)}$ are:

$$s_a^{(j)} = e_a^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{m} e_{a-z}^{(i)} g_{k+j,z}^{(i)} \qquad /1 \leq j \leq n-k \qquad (A5.2.2a)$$

$$s_{a+1}^{(j)} = e_{a+1}^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{m} e_{a+1-z}^{(i)} g_{k+j,z}^{(i)} \qquad /1 \leq j \leq n-k \qquad (A5.2.2b)$$

. . . .

. . . .

$$s_{a+m}^{(j)} = e_{a+m}^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{m} e_{a+m-z}^{(i)} g_{k+j,z}^{(i)} \qquad /1 \leq j \leq n-k \qquad (A5.2.2c)$$

In general:

$$s_{a+x}^{(j)} = e_{a+x}^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{m} e_{a+x-z}^{(i)} g_{k+j,z}^{(i)} \qquad \Big/ \begin{matrix} 1 \leq j \leq n-k \\ 0 \leq x \leq m \end{matrix} \qquad (A5.2.2d)$$

Let $m-z = w$ and rearrange the terms:

$$s_{a+x}^{(j)} = e_{a+x}^{(k+j)} + \sum_{i=1}^{k} \sum_{w=0}^{m} g_{k+j,m-w}^{(i)} e_{a+x-m+w}^{(i)} \Big/ \begin{matrix} 1 \leq j \leq n-k \\ 0 \leq x \leq m \end{matrix} \qquad (A5.2.3)$$

The 2nd summation, in the above eqn, may be written in matrix form:

$$\sum_{w=0}^{m} g^{(i)}_{k+j,m-w} e^{(i)}_{a+x-m+w} =$$

$$= \left[g^{(i)}_{k+j,m}\ g^{(i)}_{k+j,m-1} \cdots\ g^{(i)}_{k+j,0} \right] \left[e^{(i)}_{a+x-m}\ e^{(i)}_{a+x-m+1} \cdots\ e^{(i)}_{a+x} \right]^{T} \qquad \text{(A)}$$

Comparing the generator sequence $\mathbf{g}^{(i)}_{k+j}$ (see Definition 2.5, p. 23) with the first vector, in the RHS of (A), one may see that the vector is nothing more than the generator sequence with its elements arranged in the reverse order:

$$\tilde{\mathbf{g}}^{(i)}_{k+j} \triangleq \left(g^{(i)}_{k+j,m}\ g^{(i)}_{k+j,m-1} \cdots\ g^{(i)}_{k+j,0} \right) \qquad \text{(A5.2.4)}$$

Then, (A) may be written as:

$$\sum_{w=0}^{m} g^{(i)}_{k+j,m-w} e^{(i)}_{a+x-m+w} = \tilde{\mathbf{g}}^{(i)}_{k+j} \left[e^{(i)}_{a+x-m}\ e^{(i)}_{a+x-m+1} \cdots\ e^{(i)}_{a+x} \right]^{T} \qquad \text{(B)}$$

Using (B) in (A5.2.3), for $1 \le j \le n-k$ & $0 \le x \le m$:

$$s^{(j)}_{a+x} + e^{(k+j)}_{a+x} = \sum_{i=1}^{k} \tilde{\mathbf{g}}^{(i)}_{k+j} \left[e^{(i)}_{a+x-m}\ e^{(i)}_{a+x-m+1} \cdots\ e^{(i)}_{a+x} \right]^{T} \qquad \text{(A5.2.5)}$$

$$= \left[\tilde{\mathbf{g}}^{(1)}_{k+j}, \tilde{\mathbf{g}}^{(2)}_{k+j}, \ldots, \tilde{\mathbf{g}}^{(k)}_{k+j} \right] \left[e^{(1)}_{a+x-m}\ e^{(1)}_{a+x-m+1} \cdots\ e^{(1)}_{a+x}\ e^{(2)}_{a+x-m}\ e^{(2)}_{a+x-m+1} \cdots\ e^{(2)}_{a+x} \cdots \right.$$

$$\left. \cdots\ e^{(k)}_{a+x-m}\ e^{(k)}_{a+x-m+1} \cdots\ e^{(k)}_{a+x} \right]^{T} \qquad \text{(A5.2.6)}$$

In (A5.2.6) the g-coefficient vector is independent of x. The error vector may also be made to be independent of x, if it is allowed to vary between its maximum & minimum values (0 & m). Then, the error vector will be made of the k error bits of blocks $a-m, a-m+1, \ldots, a+m$. In such a case, the vector of g-coefficients must be modified, by interspersing 0s in between the $\tilde{\mathbf{g}}^{(i)}_{k+j}$s.

In (A5.2.6), the 1st bit of $\tilde{\mathbf{g}}^{(1)}_{k+j}$ multiplies $e^{(1)}_{a+x-m}$. If the 1st error bit of the error vector is $e^{(1)}_{a-m}$, then $e^{(1)}_{a+x-m}$ will be the (x+1)th bit of this vector, hence x 0s must precede $\tilde{\mathbf{g}}^{(1)}_{k+j}$. Similarly, the last bit of $\tilde{\mathbf{g}}^{(1)}_{k+j}$ multiplies $e^{(1)}_{a+x}$ and the next g-coefficient [the 1st bit of $\tilde{\mathbf{g}}^{(2)}_{k+j}$] multiplies $e^{(2)}_{a+x-m}$, while in the modified error vector, m error bits $\{e^{(1)}_{a+x+1}, e^{(1)}_{a+x+2}, \ldots, e^{(1)}_{a+m}, e^{(2)}_{a-m}, e^{(2)}_{a-m+1}, \ldots, e^{(2)}_{a+x-m-1}\}$, will be placed in between. Hence, m 0s must be placed in between $\tilde{\mathbf{g}}^{(1)}_{k+j}$ & $\tilde{\mathbf{g}}^{(2)}_{k+j}$. For the same reason, m 0s must be placed between any two of $\tilde{\mathbf{g}}^{(i)}_{k+j}$ & $\tilde{\mathbf{g}}^{(i+1)}_{k+j}$. Finally, m-x 0s must be placed after $\tilde{\mathbf{g}}^{(k)}_{k+j}$, so that the vector has the appropriate dimensions. From the last eqn:

$$s^{(j)}_{\alpha+x} + e^{(k+j)}_{\alpha+x} = \left[0_x, \breve{g}^{(1)}_{k+j}, 0_m, \breve{g}^{(2)}_{k+j}, 0_m, \ldots, 0_m, \breve{g}^{(k)}_{k+j}, 0_{m-x}\right]\left[e^{(1)}_{\alpha-m}\; e^{(1)}_{\alpha-m+1} \cdot\cdot \right.$$

$$\left. \cdot\cdot\; e^{(1)}_{\alpha}\; e^{(2)}_{\alpha-m}\; e^{(2)}_{\alpha-m+1} \cdots e^{(2)}_{\alpha} \cdots e^{(k)}_{\alpha-m}\; e^{(k)}_{\alpha-m+1} \cdots e^{(k)}_{\alpha}\right]^T \qquad (A5.2.7)$$

Consider now the following notation:

$$\left[S\right]^{\beta}_{\alpha} \triangleq \left[s^{(1)}_{\alpha}\; s^{(1)}_{\alpha+1} \cdots s^{(1)}_{\beta}\; s^{(2)}_{\alpha}\; s^{(2)}_{\alpha+1} \cdots s^{(2)}_{\beta} \cdot\cdot \right.$$

$$\left. \cdot\cdot\; s^{(n-k)}_{\alpha}\; s^{(n-k)}_{\alpha+1} \cdots s^{(n-k)}_{\beta}\right] \qquad (A5.2.8a)$$

$$\left[E^{m}\right]^{\beta}_{\alpha} \triangleq \left[e^{(1)}_{\alpha}\; e^{(1)}_{\alpha+1} \cdots e^{(1)}_{\beta}\; e^{(2)}_{\alpha}\; e^{(2)}_{\alpha+1} \cdots e^{(2)}_{\beta} \cdot\cdot \right.$$

$$\left. \cdot\cdot\; e^{(k)}_{\alpha}\; e^{(k)}_{\alpha+1} \cdots e^{(k)}_{\beta}\right] \qquad (A5.2.8b)$$

$$\left[E^{p}\right]^{\beta}_{\alpha} \triangleq \left[e^{(k+1)}_{\alpha}\; e^{(k+1)}_{\alpha+1} \cdots e^{(k+1)}_{\beta}\; e^{(k+2)}_{\alpha}\; e^{(k+2)}_{\alpha+1} \cdots e^{(k+2)}_{\beta} \cdot\cdot \right.$$

$$\left. \cdot\cdot\; e^{(n)}_{\alpha}\; e^{(n)}_{\alpha+1} \cdots e^{(n)}_{\beta}\right] \qquad (A5.2.8c)$$

From (A5.2.8b) & (A5.2.7), for $1 \le j \le n-k$ & $0 \le x \le m$::

$$s^{(j)}_{\alpha+x} + e^{(k+j)}_{\alpha+x} = \left[0_x, \breve{g}^{(1)}_{k+j}, 0_m, \breve{g}^{(2)}_{k+j}, \ldots, 0_m, \breve{g}^{(k)}_{k+j}, 0_{m-x}\right]\left\{\left[E^{m}\right]^{\alpha+m}_{\alpha-m}\right\}^T \qquad (A5.2.9)$$

where, 0_x is a $1 \times x$ row vector of 0s.

Eqn (A5.2.9) represents a system of m+1 eqns, for each $j=1,2,\ldots,n-k$:

$$\begin{vmatrix} \vdots \\ s^{(j)}_{\alpha} \\ s^{(j)}_{\alpha+1} \\ \vdots \\ s^{(j)}_{\alpha+m} \\ \vdots \end{vmatrix} = \begin{vmatrix} \cdots & & & & & & & & \cdots \\ 0_0 & \breve{g}^{(1)}_{k+j} & 0_m & \breve{g}^{(2)}_{k+j} & 0_m & \cdots & 0_m & \breve{g}^{(k)}_{k+j} & 0_m \\ 0_1 & \breve{g}^{(1)}_{k+j} & 0_m & \breve{g}^{(2)}_{k+j} & 0_m & \cdots & 0_m & \breve{g}^{(k)}_{k+j} & 0_{m-1} \\ \cdots & & & & & & & & \cdots \\ 0_m & \breve{g}^{(1)}_{k+j} & 0_m & \breve{g}^{(2)}_{k+j} & 0_m & \cdots & 0_m & \breve{g}^{(k)}_{k+j} & 0_0 \\ \cdots & & & & & & & & \cdots \end{vmatrix} E + \begin{vmatrix} \vdots \\ e^{(k+j)}_{\alpha} \\ e^{(k+j)}_{\alpha+1} \\ \vdots \\ e^{(k+j)}_{\alpha+m} \\ \vdots \end{vmatrix} \qquad (A5.2.10)$$

In the above matrix eqn (of which only a part is shown), 'big' E represents the error vector of eqn (A5.2.9). The system matrix (of which only a part is shown) may be suitably partitioned in *parity squares*. Each parity square is made of a column of m+1 $\breve{g}^{(i)}_{k+j}$s, each of which is displaced to the right (with respect to the one above) by one bit, the 'gaps' being filled by 0s. For $i=1,2,\ldots,k$ & $j=1,2,\ldots,n-k$:

$$\Gamma_i^j \;\hat{=}\; \begin{bmatrix} 0_0 & \tilde{g}_{k+j}^{(i)} & 0_m \\ 0_1 & \tilde{g}_{k+j}^{(i)} & 0_{m-1} \\ \vdots & \vdots & \vdots \\ 0_m & \tilde{g}_{k+j}^{(i)} & 0_0 \end{bmatrix} \qquad \text{(A5.2.11)}$$

Using the definition of parity squares [(A5.2.11)], in the last (incomplete) matrix eqn [(A2.5.10)]:

$$\begin{bmatrix} s_a^{(1)} \\ s_{a+1}^{(1)} \\ \cdots \\ s_{a+m}^{(1)} \\ s_a^{(2)} \\ s_{a+1}^{(2)} \\ \cdots \\ s_{a+m}^{(2)} \\ \vdots \\ s_a^{(n-k)} \\ s_{a+1}^{(n-k)} \\ \cdots \\ s_{a+m}^{(n-k)} \end{bmatrix} = \begin{bmatrix} \Gamma_1^1 & \Gamma_2^1 & \cdots & \Gamma_k^1 \\ \Gamma_1^2 & \Gamma_2^2 & \cdots & \Gamma_k^2 \\ \vdots & \vdots & \vdots & \vdots \\ \Gamma_1^{n-k} & \Gamma_2^{n-k} & \cdots & \Gamma_k^{n-k} \end{bmatrix} \begin{bmatrix} e_{a-m}^{(1)} \\ e_{a-m+1}^{(1)} \\ \cdots \\ e_{a+m}^{(1)} \\ e_{a-m}^{(2)} \\ e_{a-m+1}^{(2)} \\ \cdots \\ e_{a+m}^{(2)} \\ \vdots \\ e_{a-m}^{(k)} \\ e_{a-m+1}^{(k)} \\ \cdots \\ e_{a+m}^{(k)} \end{bmatrix} + \begin{bmatrix} e_a^{(k+1)} \\ e_{a+1}^{(k+1)} \\ \cdots \\ e_{a+m}^{(k+1)} \\ e_a^{(k+2)} \\ e_{a+1}^{(k+2)} \\ \cdots \\ e_{a+m}^{(k+2)} \\ \vdots \\ e_a^{(n)} \\ e_{a+1}^{(n)} \\ \cdots \\ e_{a+m}^{(n)} \end{bmatrix} \qquad \text{(A5.2.12a)}$$

Matrix eqn (A5.2.12a) can (and needs to) be written in a more compact form. If notation (A5.2.8) is used and $H(\Gamma)$ denotes the system matrix in eqn (A5.2.12a), then the latter can be written as:

$$\left\{\left[S\right]_{a}^{a+m}\right\}^{T} = H(\Gamma)\left\{\left[E^{m}\right]_{a-m}^{a+m}\right\}^{T} + \left\{\left[E^{p}\right]_{a}^{a+m}\right\}^{T} \qquad (A5.2.12b)$$

If the transpose of both sides in the last eqn is obtained (see, also, Theorem A2.2.1, p. 300), then:

$$\left[S\right]_{a}^{a+m} = \left[E^{m}\right]_{a-m}^{a+m}\left[H(\Gamma)\right]^{T} + \left[E^{p}\right]_{a}^{a+m} \qquad (A5.2.12c)$$

The dimensions of the matrices in eqn (A5.2.12c) are as following:

$\left[S\right]_{a}^{a+m}$ is a 1 x (n-k)(m+1) matrix,

$\left[E^{m}\right]_{a-m}^{a+m}$ is a 1 x k(2m+1) matrix,

$\left[H(\Gamma)\right]^{T}$ is a k(2m+1) x (n-k)(m+1) matrix and

$\left[E^{p}\right]_{a}^{a+m}$ is a 1 x (n-k)(m+1) matrix.

Consider any error bit, say, $e_{a-\beta}^{(\mu)}$ $/-m\leq\beta\leq m$ & $1\leq\mu\leq k$ and examine if any particular syndrome bit, say, $s_{a+\tau}^{(\sigma)}$ $/0\leq\tau\leq m$, $1\leq\sigma\leq n-k$ checks on it. By inspection of eqn (A5.2.12a) one may conclude that the error bit belongs to the μth group of error bits and within this group it is the $(m+1-\beta)$th bit, i.e. it is the $[(\mu-1)(2m+1)+m+1-\beta]$th bit of the message error vector. The syndrome bit, on the other hand, belongs to the σth group of syndrome bits and within this group it is the $(\tau+1)$th bit, i.e. it is the $[(\sigma-1)(m+1)+\tau+1]$th bit of the syndrome vector. Then, these two bits are linked via the $[(\sigma-1)(m+1)+\tau+1]$th row, $[(\mu-1)(2m+1)+m+1-\beta]$th column, g-coefficient of the system matrix $H(\Gamma)$. Since the latter is organized in an (n-k) x k matrix of Γs, each of which is an (m+1) x (2m+1) matrix [see (A5.2.11)], the above mentioned g-coefficient belongs to the σth row of Γs and μth column of Γs, i.e. to Γ_{μ}^{σ}, which contains shifted versions of $\mathfrak{g}_{k+\sigma}^{(\mu)}$. Within this parity square, the g-coefficient belongs to the $(\tau+1)$th row, $(m+1-\beta)$th column. If one expands the parity square, one may see that $g_{k+\sigma,z}^{(\mu)}$ is found in rows & columns satisfying: z = = m + row - column. Hence, $z = m + (\tau+1) - (m+1-\beta) = \tau+\beta$. Hence, if $-m\leq\beta\leq m$, $1\leq\mu\leq k$, $0\leq\tau\leq m$ & $1\leq\sigma\leq n-k$, then:

$$s_{a+\tau}^{(\sigma)} \text{ checks on } e_{a-\beta}^{(\mu)} \text{ iff } g_{k+\sigma,\tau+\beta}^{(\mu)} = 1 \qquad (A5.2.13)$$

Also, from the discussion preceding reln (A5.2.13), the syndromes checking on error bit $e_{a-\beta}^{(\mu)}$ are determined by the

$[(\mu-1)(2m+1)+m+1-\beta]$th column of $H(\Gamma)$; in particular, the 1s along this column indicate the positions, within the syndrome vector, of the syndrome bits checking on $e^{(\mu)}_{\alpha-\beta}$ (the top bit is in position 1). Similarly, the message error bits checked by $s^{(\sigma)}_{\alpha+\tau}$ are determined by the $[(\sigma-1)(m+1)+\tau+1]$th row, of $H(\Gamma)$; in particular, the 1s along this row indicate the positions, within the message error vector, of the error bits checked by $s^{(\sigma)}_{\alpha+\tau}$.

Consider now the problem of determining $J_{\mu,\alpha}$, the number of syndrome bits checking on error bit $e^{(\mu)}_{\alpha}$. Matrix equation (A5.2.12) contains, by design, all the syndromes that check on this bit. By (A5.2.13), for $\beta=0$, the number of syndromes checking on this bit equals the number of $g^{(\mu)}_{k+\sigma,\tau}$s that are equal to one, where μ is fixed, but σ & τ are allowed to vary over their range. Hence:

$$J_{\mu,\alpha} = \sum_{\sigma=1}^{n-k} \sum_{\tau=0}^{m} g^{(\mu)}_{k+\sigma,\tau} = \sum_{\sigma=1}^{n-k} w\left[g^{(\mu)}_{k+\sigma}\right] \qquad (A5.2.14)$$

Note, from (A5.2.14) that $J_{\mu,\alpha}$ depends only on the bit number μ (within a block) and not on the time-unit α (provided that $\alpha\geq m$, as initially assumed). Hence α may drop from $J_{\mu,\alpha}$.

Finally, the size, $c_{j,h}$, of syndrome $s^{(j)}_{h}$ $/h\geq m$ and $1\leq j\leq n-k$ may be calculated from eqn (A5.2.1). Because $h\geq m$, then $\theta=m$ and the size of the syndrome bit equals the number of g-coefficients that are equal to 1, plus 1 (for the parity-check):

$$c_{j,h} = 1 + \sum_{i=1}^{k} \sum_{z=0}^{m} g^{(i)}_{k+j,z} = 1 + \sum_{i=1}^{k} w\left[g^{(i)}_{k+j}\right] \qquad (A5.2.15)$$

Note, again, that $c_{j,h}$ depends only on j, so h may drop.

The following theorem has been proved:

<u>Theorem A5.2.1</u>: Consider an (n,k,m) systematic convolutional code with generator sequences $g^{(i)}_{k+j}$ $/i=1,2,..,k$ & $j=1,2,...,n-k$. Then, under definite decoding, for $\alpha\geq0$:

$$\left[S\right]_{\alpha}^{\alpha+m} = \left[E^{m}\right]_{\alpha-m}^{\alpha+m}\left[H(\Gamma)\right]^{T} + \left[E^{p}\right]_{\alpha}^{\alpha+m} \qquad \text{(A5.2.12c)}$$

where if $\alpha<m$, the message error vector is suitably truncated. If $-m\le\beta\le m$, $1\le\mu\le k$, $0\le\tau\le m$ & $1\le\sigma\le n-k$, then:

$$s_{\alpha+\tau}^{(\sigma)} \text{ checks on } e_{\alpha-\beta}^{(\mu)} \text{ iff } g_{k+\sigma,\tau+\beta}^{(\mu)} = 1 \qquad \text{(A5.2.13)}$$

The syndromes checking on $e_{\alpha-\beta}^{(\mu)}$, correspond to 1s along the $[(\mu-1)(2m+1)+m+1-\beta]$th column of $H(\Gamma)$. The message error bits checked by syndrome bit $s_{\alpha+\tau}^{(\sigma)}$, correspond to 1s along the $[(\sigma-1)(m+1)+\tau+1]$th row, of $H(\Gamma)$.

Furthermore, if J_i /$1\le i\le k$ denotes the number of syndromes checking on error bit $e_h^{(i)}$ /$h\ge m$ and c_j /$1\le j\le n-k$ denotes the size of syndrome bit $s_h^{(j)}$ /$h\ge m$, then:

$$J_i = \sum_{j=1}^{n-k} w\left[g_{k+j}^{(i)}\right] \qquad /1\le i\le k \qquad \text{(A5.2.14)}$$

$$c_j = 1 + \sum_{i=1}^{k} w\left[g_{k+j}^{(i)}\right] \qquad /1\le j\le n-k \qquad \text{(A5.2.15)}$$

∎

Consider an example:

<u>Example A5.2.1</u>: Consider the (2,1,6) systematic code with generator polynomial $g_2^{(1)}(D) = 1+D^2+D^5+D^6$. Since $n-k = 1$ there is only one syndrome bit, $s_\alpha^{(1)}$ /$\alpha\ge0$, which is related to the error bits via matrix eqn (A5.2.12a). Furthermore, there is only one parity square, Γ_1^1, hence this coincides with the system matrix $H(\Gamma)$.

Let us consider the application of the results of Theorem A5.2.1, to the above example:

According to reln (A5.2.13), $s_{\alpha+\tau}^{(\sigma)}$ checks on $e_{\alpha-\beta}^{(\mu)}$, iff $g_{1+\sigma,\tau+\beta}^{(\mu)} = 1$, where $-6\le\beta\le6$, $1\le\mu\le1$, $0\le\tau\le6$ and $1\le\sigma\le2-1$. Hence, $s_{\alpha+\tau}^{(1)}$ checks on $e_{\alpha-\beta}^{(1)}$ if, and only if, $g_{2,\tau+\beta}^{(1)} = 1$, where $-6\le\beta\le6$ & $0\le\tau\le6$. From the given generator polynomial:

$$g_{2,0}^{(1)} = g_{2,2}^{(1)} = g_{2,5}^{(1)} = g_{2,6}^{(1)} = 1$$

$$g_{2,1}^{(1)} = g_{2,3}^{(1)} = g_{2,4}^{(1)} = 0$$

From above: For, say, $\beta=2$ & $\tau=5$, $g_{2,5+2}^{(1)} = 0$, hence $s_{\alpha+5}^{(1)}$ does not check on $e_{\alpha-2}^{(1)}$. For, say, $\beta=2$ & $\tau=0$, $g_{2,0+2}^{(1)} = 1$, hence $s_\alpha^{(1)}$

does check on $e^{(1)}_{a-2}$. For, say, $\beta=-1$ & $\tau=5$, $g^{(1)}_{2,5-1} = 0$, hence $s^{(1)}_{a+5}$ does not check on $e^{(1)}_{a+1}$. For, say, $\beta=-5$ & $\tau=5$, $g^{(1)}_{2,5-5} = 1$, hence $s^{(1)}_{a+5}$ does check on $e^{(1)}_{a+5}$. These 'predictions' can be verified from the matrix eqn, below.

Furthermore, J_1 denotes the number of syndromes checking on error bit $e^{(1)}_a$ /$a \geq 6$ and c_1 denotes the size of syndrome bit $s^{(1)}_a$ /$a \geq 6$. From (A5.2.14) and (A5.2.15), and since the weight of the, only, generator sequence is 4: $J_1 = 4$ & $c_1 = 5$, which may be verified from the matrix eqn, below.

Eqn (A5.2.12a) gives:

$$\begin{bmatrix} s^{(1)}_a \\ s^{(1)}_{a+1} \\ s^{(1)}_{a+2} \\ s^{(1)}_{a+3} \\ s^{(1)}_{a+4} \\ s^{(1)}_{a+5} \\ s^{(1)}_{a+6} \end{bmatrix} = \begin{bmatrix} 1&1&0&0&1&0&1&&&&&& \\ &1&1&0&0&1&0&1&&&&& \\ &&1&1&0&0&1&0&1&&&& \\ &&&1&1&0&0&1&0&1&&& \\ &&&&1&1&0&0&1&0&1&& \\ &&&&&1&1&0&0&1&0&1& \\ &&&&&&1&1&0&0&1&0&1 \end{bmatrix} \begin{bmatrix} e^{(1)}_{a-6} \\ e^{(1)}_{a-5} \\ e^{(1)}_{a-4} \\ e^{(1)}_{a-3} \\ e^{(1)}_{a-2} \\ e^{(1)}_{a-1} \\ e^{(1)}_{a} \\ e^{(1)}_{a+1} \\ e^{(1)}_{a+2} \\ e^{(1)}_{a+3} \\ e^{(1)}_{a+4} \\ e^{(1)}_{a+5} \\ e^{(1)}_{a+6} \end{bmatrix} + \begin{bmatrix} e^{(2)}_a \\ e^{(2)}_{a+1} \\ e^{(2)}_{a+2} \\ e^{(2)}_{a+3} \\ e^{(2)}_{a+4} \\ e^{(2)}_{a+5} \\ e^{(2)}_{a+6} \end{bmatrix}$$

APPENDIX 5.3: FEEDBACK DECODING - PARITY TRIANGLES

For FD, it is enough to consider the decoding of r_0, since all subsequent blocks are decoded in exactly the same way (assuming no error propagation). Consider the following rearranged version of eqns (5.8).

$$s_0^{(j)} = e_0^{(k+j)} + \sum_{i=1}^{k} e_0^{(i)} g_{k+j,0}^{(i)} \qquad /1 \le j \le n-k \qquad \text{(A5.3.1a)}$$

$$s_1^{(j)} = e_1^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{1} e_z^{(i)} g_{k+j,1-z}^{(i)} \qquad /1 \le j \le n-k \qquad \text{(A5.3.1b)}$$

. . . .

. . . .

$$s_m^{(j)} = e_m^{(k+j)} + \sum_{i=1}^{k} \sum_{z=0}^{m} e_z^{(i)} g_{k+j,m-z}^{(i)} \qquad /1 \le j \le n-k \qquad \text{(A5.3.1c)}$$

Consider the expansion of the above eqns (i=1,2,..,k):

$$s_0^{(j)} = \cdots + g_{k+j,0}^{(i)} e_0^{(i)} + \cdots \qquad \cdots + e_0^{(k+j)}$$

$$s_1^{(j)} = \cdots + g_{k+j,1}^{(i)} e_0^{(i)} + g_{k+j,0}^{(i)} e_1^{(i)} + \cdots \qquad \cdots + e_1^{(k+j)}$$

. . . .

. . . .

. . . .

$$s_m^{(j)} = \cdots + g_{k+j,m}^{(i)} e_0^{(i)} + g_{k+j,m-1}^{(i)} e_1^{(i)} + \cdots + g_{k+j,0}^{(i)} e_m^{(i)} + \cdots \quad \cdots + e_m^{(k+j)}$$

The above eqns can be written in matrix form:

$$\begin{vmatrix} \cdots \\ s_0^{(j)} \\ s_1^{(j)} \\ \vdots \\ s_m^{(j)} \\ \cdots \end{vmatrix} = \begin{vmatrix} \cdots & & & & \cdots \\ \cdots & g_{k+j,0}^{(i)} & & & \cdots \\ \cdots & g_{k+j,1}^{(i)} & g_{k+j,0}^{(i)} & & \cdots \\ \vdots & & & \ddots & \\ \cdots & g_{k+j,m}^{(i)} & g_{k+j,m-1}^{(i)} \cdots & g_{k+j,0}^{(i)} & \cdots \\ \cdots & & & & \cdots \end{vmatrix} \begin{vmatrix} \cdots \\ e_0^{(i)} \\ e_1^{(i)} \\ \vdots \\ e_m^{(i)} \\ \cdots \end{vmatrix} + \begin{vmatrix} \cdots \\ e_0^{(k+j)} \\ e_1^{(k+j)} \\ \vdots \\ e_m^{(k+j)} \\ \cdots \end{vmatrix} \qquad \text{(A5.3.2)}$$

Note that in eqn (A5.3.2) the spaces denote zeros. It is obvious that the matrix of the generator coefficients is made of 'triangles' of gs.

<u>Definition A5.3.1</u>: The general jith *parity triangle* where i=1,2,...,k & j=1,2,...,n-k, is defined as following:

$$\Delta_i^j \triangleq \begin{bmatrix} g^{(i)}_{k+j,0} & & & \\ g^{(i)}_{k+j,1} & g^{(i)}_{k+j,0} & & \\ \cdots & \cdots & \cdots & \\ g^{(i)}_{k+j,m} & g^{(i)}_{k+j,m-1} & \cdots & g^{(i)}_{k+j,0} \end{bmatrix} \tag{A5.3.3}$$

∎

Eqn (A5.3.2) can be written as following:

$$\begin{bmatrix} s_0^{(1)} \\ s_1^{(1)} \\ \cdots \\ s_m^{(1)} \\ s_0^{(2)} \\ s_1^{(2)} \\ \cdots \\ s_m^{(2)} \\ \cdots \\ \cdots \\ \cdots \\ s_0^{(n-k)} \\ s_1^{(n-k)} \\ \cdots \\ s_m^{(n-k)} \end{bmatrix} = \begin{bmatrix} \Delta_1^1 & \Delta_2^1 & \cdots & \Delta_k^1 \\ \Delta_1^2 & \Delta_2^2 & \cdots & \Delta_k^2 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ \Delta_1^{n-k} & \Delta_2^{n-k} & \cdots & \Delta_k^{n-k} \end{bmatrix} \begin{bmatrix} e_0^{(1)} \\ e_1^{(1)} \\ \cdots \\ e_m^{(1)} \\ e_0^{(2)} \\ e_1^{(2)} \\ \cdots \\ e_m^{(2)} \\ \cdots \\ \cdots \\ \cdots \\ e_0^{(k)} \\ e_1^{(k)} \\ \cdots \\ e_m^{(k)} \end{bmatrix} + \begin{bmatrix} e_0^{(k+1)} \\ e_1^{(k+1)} \\ \cdots \\ e_m^{(k+1)} \\ e_0^{(k+2)} \\ e_1^{(k+2)} \\ \cdots \\ e_m^{(k+2)} \\ \cdots \\ \cdots \\ \cdots \\ e_0^{(n)} \\ e_1^{(n)} \\ \cdots \\ e_m^{(n)} \end{bmatrix} \tag{A5.3.4a}$$

Matrix eqn (A5.3.4a) can (and needs to) be written in a more compact form. Using the notation introduced by (A5.2.8) and letting $H(\natural)$ denote the system matrix:

$$\left\{\left[S\right]_0^m\right\}^T = H(\natural)\left\{\left[E^m\right]_0^m\right\}^T + \left\{\left[E^p\right]_0^m\right\}^T \qquad \text{(A5.3.4b)}$$

If the transpose of both sides in the last eqn is obtained (see, also, Theorem A2.2.1, p. 300), then:

$$\left[S\right]_0^m = \left[E^m\right]_0^m\left[H(\natural)\right]^T + \left[E^p\right]_0^m \qquad \text{(A5.3.4c)}$$

The dimensions of the matrices in eqn (A5.3.4c) are as following:

$\left[S\right]_0^m$ is a 1 x (n-k)(m+1) matrix,

$\left[E^m\right]_0^m$ is a 1 x k(m+1) matrix,

$\left[H(\natural)\right]^T$ is a k(m+1) x (n-k)(m+1) matrix and

$\left[E^p\right]_0^m$ is a 1 x (n-k)(m+1) matrix.

From the definition of the parity triangle [eqn (A5.3.3)], one may conclude that it is made of (m+1)(m+2)/2 g-coefficients. Furthermore, a comparison of the 1st column of the parity triangle with the coefficients of the generator sequence $g_{k+j}^{(i)}$ (see Definition 2.5), reveals that they are identical. Also, if the 1st column is shifted downwards by one position and truncated in the bottom-end (by one element) the 2nd column is obtained. In fact every column is a shifted/truncated version of the 1st. The following note summarizes the findings.

<u>Note A5.3.1</u>: Consider an (n,k,m) systematic convolutional code. This code has k(n-k) parity triangles, each of which contains (m+1)(m+2)/2 elements. The 1st column of parity triangle $\natural_i^j$ (i=1,2,..,k & j=1,2,..,n-k) is $[g_{k+j}^{(i)}]^T$, i.e. the transpose of the (i,k+j)th generator sequence [or, the same, the column of the elements of the (i,k+j)th generator polynomial, $g_{k+j}^{(i)}(D)$]. The hth column ($1 \le h \le m+1$) of the triangle is obtained by a downward shift of the 1st column by h-1 positions and a truncation of its bottom end, by h-1 elements.

∎

Consider now any error bit, say, $e_{\alpha}^{(\mu)}$ /$0\le\alpha\le m$ & $1\le\mu\le k$ and examine if any particular syndrome bit, say, $s_{\tau}^{(\sigma)}$ /$0\le\tau\le m$ & $1\le\sigma\le n-k$ checks on it. By inspection of eqn (A5.3.4a) one may conclude that the error bit belongs to the μth group of error bits and within this group it is the $(\alpha+1)$th bit, i.e. it is the $[(\mu-1)(m+1)+\alpha+1]$th bit of the message error vector. The syndrome bit, on the other hand belongs to the σth group of syndrome bits and within this group it is the $(\tau+1)$th bit, i.e. it is the $[(\sigma-1)(m+1)+\tau+1]$th bit of the syndrome vector. Then, these two bits are linked via the $[(\sigma-1)(m+1)+\tau+1]$th row, $[(\mu-1)(m+1)+\alpha+1]$th column, g-coefficient of the system matrix $H(\Delta)$. Since the latter is organized in an $(n-k) \times k$ matrix of Δs, each of which is an $(m+1) \times (m+1)$ matrix [see (A5.3.3)], the above mentioned g-coefficient belongs to the σth row of Δs and μth column of Δs, i.e. to Δ_{μ}^{σ}, which contains shifted/truncated versions of $g_{k+\sigma}^{(\mu)}$. Within this parity triangle, the g-coefficient belongs to the $(\tau+1)$th row, $(\alpha+1)$th column. If one expands the parity triangle, one may see that $g_{k+\sigma,z}^{(\mu)}$ is found in rows and columns satisfying: z = row - column. So, $z = (\tau+1) - (\alpha+1) = \tau-\alpha$. Hence, if $0\le\alpha\le m$, $1\le\mu\le k$, $0\le\tau\le m$ & $1\le\sigma\le n-k$, then:

$$s_{\tau}^{(\sigma)} \text{ checks on } e_{\alpha}^{(\mu)} \text{ iff } g_{k+\sigma,\tau-\alpha}^{(\mu)} = 1 \qquad \text{(A5.3.5)}$$

Also, from the discussion preceding reln (A5.3.5), the syndromes checking on error bit $e_{\alpha}^{(\mu)}$ are determined by the $[(\mu-1)(m+1)+\alpha+1]$th column of $H(\Delta)$; in particular, the 1s along this column indicate the positions, within the syndrome vector, of the syndrome bits checking on $e_{\alpha}^{(\mu)}$ (the top bit is in position 1). Similarly, the bits from the message error vector, checked by syndrome bit $s_{\tau}^{(\sigma)}$, are determined by the $[(\sigma-1)(m+1)+\tau+1]$th row, of $H(\Delta)$; in particular, the 1s along this row indicate the positions, within the message error vector, of the error bits checked by $s_{\tau}^{(\sigma)}$.

Consider now the problem of determining $J_{\mu,0}$, the number of syndrome bits checking on error bit $e_{0}^{(\mu)}$. Matrix equation (A5.3.4a) contains, by design, all the syndromes that check on this bit. By (A5.3.5), for $\alpha=0$, the number of syndromes checking on this bit equals the number of $g_{k+\sigma,\tau}^{(\mu)}$s that are equal to one, where μ is fixed, but σ & τ are allowed to vary over their range. Hence:

$$J_{\mu,0} = \sum_{\sigma=1}^{n-k} \sum_{\tau=0}^{m} g_{k+\sigma,\tau}^{(\mu)} = \sum_{\sigma=1}^{n-k} w\left[g_{k+\sigma}^{(\mu)}\right] \qquad (A5.3.6)$$

Note, from (A5.3.6) that $J_{\mu,0}$ depends only on the bit number μ (within a block) and not on the time-unit 0. Hence 0 may drop from $J_{\mu,0}$. In any case, the equations for the decoding of the zeroth block are identical to those for any other block.

Finally, the size, $c_{j,h}$ (/h≥0 & 1≤j≤n-k), of syndrome bit $s_h^{(j)}$ may be calculated from eqn (5.7). Again, under FD, the decoding of the 0th block is similar to that of any other block. Hence, the syndromes checking on any block are the same linear combinations of error bits, like the syndromes used for the decoding of r_0. Then, h≤m and hence, $\theta \hat{=} \mathrm{MIN}\{h,m\} = h$. The size of the syndrome equals the number of g-coefficients that are equal to 1, plus 1 (for the parity-check):

$$c_{j,h} = 1 + \sum_{i=1}^{k} \sum_{z=0}^{h} g_{k+j,z}^{(i)} \qquad (A5.3.7)$$

The following theorem has been proved:

<u>Theorem A5.3.1</u>: Consider an (n,k,m) systematic convolutional code with generator sequences $g_{k+j}^{(i)}$ /i=1,2,...,k & j=1, 2,...,n-k. Then, under feedback decoding:

$$\left[S\right]_0^m = \left[E^m\right]_0^m \left[H(\mathsf{k})\right]^T + \left[E^p\right]_0^m \qquad (A5.3.4c)$$

If 0≤α≤m, 1≤μ≤k, 0≤τ≤m & 1≤σ≤n-k, then:

$$s_\tau^{(\sigma)} \text{ checks on } e_\alpha^{(\mu)} \text{ iff } g_{k+\sigma,\tau-\alpha}^{(\mu)} = 1 \qquad (A5.3.5)$$

The syndrome bits checking on $e_\alpha^{(\mu)}$ correspond to 1s along the [(μ-1)(m+1)+α+1]th column of $H(\mathsf{k})$. The message error bits checked by syndrome bit $s_\tau^{(\sigma)}$ correspond to 1s along the [(σ-1)(m+1)+τ+1]th row, of $H(\mathsf{k})$.

Furthermore, if J_i /1≤i≤k denotes the number of syndromes checking on error bit $e_0^{(i)}$ and $c_{j,h}$ /1≤j≤n-k denotes the size of syndrome bit $s_h^{(j)}$ /h≥0, then:

$$J_i = \sum_{j=1}^{n-k} w\left[g_{k+j}^{(i)}\right] \quad /1 \le i \le k \qquad (A5.3.6)$$

$$c_{j,h} = 1 + \sum_{i=1}^{k} \sum_{z=0}^{h} g_{k+j,z}^{(i)} \quad /1 \le j \le n-k \qquad (A5.3.7)$$

■

Consider the following two examples:

<u>Example A5.3.1</u>: Consider the (2,1,6) systematic convolutional code with generator polynomial $g_2^{(1)}(D) = 1+D^2+D^5+D^6$ (examined, under DD, in Example A5.2.1). Since n-k = 1 there is only one syndrome bit, $s_a^{(1)}$ /a≥0, which is related to the error bits via matrix eqn (A5.3.4a). Furthermore, there is only one parity triangle, Δ_1^1, hence it coincides with the system matrix $H(\Delta)$.

Eqn (A5.3.4a) gives:

$$\begin{bmatrix} s_0^{(1)} \\ s_1^{(1)} \\ s_2^{(1)} \\ s_3^{(1)} \\ s_4^{(1)} \\ s_5^{(1)} \\ s_6^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & & & & & & \\ 0 & 1 & & & & & \\ 1 & 0 & 1 & & & & \\ 0 & 1 & 0 & 1 & & & \\ 0 & 0 & 1 & 0 & 1 & & \\ 1 & 0 & 0 & 1 & 0 & 1 & \\ 1 & 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} e_0^{(1)} \\ e_1^{(1)} \\ e_2^{(1)} \\ e_3^{(1)} \\ e_4^{(1)} \\ e_5^{(1)} \\ e_6^{(1)} \end{bmatrix} + \begin{bmatrix} e_0^{(2)} \\ e_1^{(2)} \\ e_2^{(2)} \\ e_3^{(2)} \\ e_4^{(2)} \\ e_5^{(2)} \\ e_6^{(2)} \end{bmatrix}$$

Let us consider the application of the results of Theorem A5.3.1, to the above example:

According to relation (A5.3.5), $s_\tau^{(\sigma)}$ checks on $e_a^{(\mu)}$, iff $g_{k+\sigma,\tau-a}^{(\mu)} = 1$, where $0 \le a \le 6$, $1 \le \mu \le 1$, $0 \le \tau \le 6$ & $1 \le \sigma \le 2-1$. Hence, $s_\tau^{(1)}$ checks on $e_a^{(1)}$, iff $g_{2,\tau-a}^{(1)} = 1$, where $0 \le a \le 6$ & $0 \le \tau \le 6$. From the given generator polynomial:

$$g_{2,0}^{(1)} = g_{2,2}^{(1)} = g_{2,5}^{(1)} = g_{2,6}^{(1)} = 1$$

$$g_{2,1}^{(1)} = g_{2,3}^{(1)} = g_{2,4}^{(1)} = 0$$

From above: For, say, $\alpha=1$ & $\tau=5$, $g_{2,5-1}^{(1)} = 0$, hence $s_5^{(1)}$ does not check on $e_1^{(1)}$. For, say, $\alpha=0$ & $\tau=5$, $g_{2,5-0}^{(1)} = 1$, hence $s_5^{(1)}$ does check on $e_0^{(1)}$. For, say, $\alpha=0$ & $\tau=3$, $g_{2,3-0}^{(1)} = 0$, hence $s_3^{(1)}$ does not check on $e_0^{(1)}$. For, say, $\alpha=2$ & $\tau=2$, $g_{2,2-2}^{(1)} = 1$, hence $s_2^{(1)}$ does check on $e_2^{(1)}$. These 'predictions' can be verified from the matrix eqn, above.

Furthermore, J_1 denotes the number of syndromes checking on error bit $e_0^{(1)}$ and $c_{1,\alpha}$ /$\alpha\geq0$ denotes the size of syndrome bit $s_\alpha^{(1)}$ /$\alpha\geq0$. From (A5.3.6), and since the weight of the, only, generator sequence is 4, $J_1 = 4$ which may be verified from the matrix eqn, above.

From eqn (A5.3.7): $c_{1,h} = 1 + g_{2,0}^{(1)} + g_{2,1}^{(1)} +\cdots+ g_{2,h}^{(1)}$

So, the size of $s_0^{(1)}$ is $c_{1,0} = 1 + g_{2,0}^{(1)} = 2$

the size of $s_3^{(1)}$ is $c_{1,3} = 1 + g_{2,0}^{(1)} + g_{2,1}^{(1)} +\cdots+ g_{2,3}^{(1)} = 3$

the size of $s_5^{(1)}$ is $c_{1,5} = 1 + g_{2,0}^{(1)} + g_{2,1}^{(1)} +\cdots+ g_{2,5}^{(1)} = 4$

Again, these 'predictions' may be verified from the last matrix eqn.

∎

Example A5.3.2: Consider the (3,2,13) systematic convolutional code with generator polynomials $g_3^{(1)} = 1+D^8+D^9+D^{12}$ & $g_3^{(2)} = 1+D^6+D^{11}+D^{13}$.

Since $k = 2$ & $n-k = 1$, $H(\lambda)$ is made of two parity triangles arranged in one row, λ_1^1 & λ_2^1. Eqn (A5.3.4a) for this example, gives (see matrix eqn overleaf):

Let us consider the application of the results of Theorem A5.3.1, to the above example:

According to relation (A5.3.5), $s_\tau^{(\sigma)}$ checks on $e_\alpha^{(\mu)}$, iff $g_{k+\sigma,\tau-\alpha}^{(\mu)} = 1$, where $0\leq\alpha\leq13$, $1\leq\mu\leq2$, $0\leq\tau\leq13$ and $1\leq\sigma\leq2-1$. Hence, $s_\tau^{(1)}$ checks on $e_\alpha^{(\mu)}$, iff $g_{3,\tau-\alpha}^{(\mu)} = 1$, where $\mu=1,2$, $0\leq\alpha\leq13$ and $0\leq\tau\leq13$. From the given generator polynomials:

$$\begin{bmatrix} s_0^{(1)} \\ s_1^{(1)} \\ s_2^{(1)} \\ s_3^{(1)} \\ s_4^{(1)} \\ s_5^{(1)} \\ s_6^{(1)} \\ s_7^{(1)} \\ s_8^{(1)} \\ s_9^{(1)} \\ s_{10}^{(1)} \\ s_{11}^{(1)} \\ s_{12}^{(1)} \\ s_{13}^{(1)} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 01 & 01 \\ 001 & 001 \\ 0001 & 0001 \\ 00001 & 00001 \\ 000001 & 000001 \\ 0000001 & 1000001 \\ 00000001 & 01000001 \\ 100000001 & 001000001 \\ 1100000001 & 0001000001 \\ 01100000001 & 00001000001 \\ 001100000001 & 100001000001 \\ 1001100000001 & 0100001000001 \\ 01001100000001 & 10100001000001 \end{bmatrix} \begin{bmatrix} e_0^{(1)} \\ e_1^{(1)} \\ e_2^{(1)} \\ e_3^{(1)} \\ e_4^{(1)} \\ e_5^{(1)} \\ e_6^{(1)} \\ e_7^{(1)} \\ e_8^{(1)} \\ e_9^{(1)} \\ e_{10}^{(1)} \\ e_{11}^{(1)} \\ e_{12}^{(1)} \\ e_{13}^{(1)} \\ e_0^{(2)} \\ e_1^{(2)} \\ e_2^{(2)} \\ e_3^{(2)} \\ e_4^{(2)} \\ e_5^{(2)} \\ e_6^{(2)} \\ e_7^{(2)} \\ e_8^{(2)} \\ e_9^{(2)} \\ e_{10}^{(2)} \\ e_{11}^{(2)} \\ e_{12}^{(2)} \\ e_{13}^{(2)} \end{bmatrix} + \begin{bmatrix} e_0^{(3)} \\ e_1^{(3)} \\ e_2^{(3)} \\ e_3^{(3)} \\ e_4^{(3)} \\ e_5^{(3)} \\ e_6^{(3)} \\ e_7^{(3)} \\ e_8^{(3)} \\ e_9^{(3)} \\ e_{10}^{(3)} \\ e_{11}^{(3)} \\ e_{12}^{(3)} \\ e_{13}^{(3)} \end{bmatrix}$$

$g^{(1)}_{3,0} = g^{(1)}_{3,8} = g^{(1)}_{3,9} = g^{(1)}_{3,12} = g^{(2)}_{3,0} = g^{(2)}_{3,6} = g^{(2)}_{3,11} = g^{(2)}_{3,13} = 1$

From above: For, say, $\alpha=1$, $\mu=1$ & $\tau=5$, $g^{(1)}_{3,5-1} = 0$, hence $s^{(1)}_5$ does not check on $e^{(1)}_1$. For, say, $\alpha=2$, $\mu=2$ & $\tau=13$, $g^{(2)}_{3,13-2} = 1$, hence $s^{(1)}_{13}$ does check on $e^{(2)}_2$. For, say, $\alpha=7$, $\mu=1$ & $\tau=10$, $g^{(1)}_{3,10-7} = 0$, hence $s^{(1)}_{10}$ does not check on $e^{(1)}_7$. For, say, $\alpha=7$, $\mu=2$ & $\tau=13$, $g^{(2)}_{3,13-7} = 1$, hence $s^{(1)}_{13}$ does check on $e^{(2)}_7$. These 'predictions' can be verified from the matrix eqn, above.

Furthermore, J_i /i=1,2 denotes the number of syndromes checking on error bit $e^{(1)}_0$ and $c_{1,\alpha}$ /$\alpha\geq0$ denotes the size of syndrome bit $s^{(1)}_\alpha$ /$\alpha\geq0$. From (A5.3.6), and since the weight of each generator sequence is 4, $J_1 = J_2 = 4$ which may be verified from the matrix eqn, above.

From eqn (A5.3.7):

$$c_{1,h} = 1 + g^{(1)}_{3,0} + g^{(2)}_{3,0} + g^{(1)}_{3,1} + g^{(2)}_{3,1} +\cdots+ g^{(1)}_{3,h} + g^{(2)}_{3,h}$$

So, the size of $s^{(1)}_0$ is $c_{1,0} = 1 + g^{(1)}_{3,0} + g^{(2)}_{3,0} = 3$

the size of $s^{(1)}_6$ is

$$c_{1,6} = 1 + g^{(1)}_{3,0} + g^{(2)}_{3,0} + g^{(1)}_{3,1} + g^{(2)}_{3,1} +\cdots+ g^{(1)}_{3,6} + g^{(2)}_{3,6} = 4$$

the size of $s^{(1)}_{11}$ is

$$c_{1,11} = 1 + g^{(1)}_{3,0} + g^{(2)}_{3,0} + g^{(1)}_{3,1} + g^{(2)}_{3,1} +\cdots+ g^{(1)}_{3,11} + g^{(2)}_{3,11} = 7$$

Again, these 'predictions' may be verified from the last matrix eqn.

∎

APPENDIX 5.4: PROOF OF THE THEORY IN SECTION 5.3

A5.4.1. Preliminary Results

Theorem A5.4.1: Let two binary n-tuples $\alpha = (\alpha_1\ \alpha_2 \cdots \alpha_n)$ and $\beta = (\beta_1\ \beta_2 \cdots \beta_n)$. Then,

$$w[\alpha] + w[\beta] \geq w[\alpha\oplus\beta] \qquad (A5.4.1)$$

where $\alpha\oplus\beta$ is the 'bit-by-bit mod-2 sum' of α & β and $\alpha_i,\beta_i\in GF(2)$ /i=1,2,...,n.

Proof: Let + denote ordinary real-number addition and ⊕ denote mod-2 addition. Then, for all i=1,2,...,n:

$$\alpha_i + \beta_i \;\left[\begin{array}{ll} 0 & \text{if } \alpha_i = \beta_i = 0 \\ 1 & \text{if } \alpha_i \neq \beta_i \\ 2 & \text{if } \alpha_i = \beta_i = 1 \end{array}\right] \longrightarrow \qquad \text{(A)}$$

$$\alpha_i \oplus \beta_i \;\left[\begin{array}{ll} 0 & \text{if } \alpha_i = \beta_i = 0 \\ 1 & \text{if } \alpha_i \neq \beta_i \\ 0 & \text{if } \alpha_i = \beta_i = 1 \end{array}\right] \longrightarrow \qquad \text{(B)}$$

From (A) & (B):

$$\alpha_i \oplus \beta_i \le \alpha_i + \beta_i \quad /i=1,2,\ldots,n \qquad \text{(A5.4.2)}$$

$$\longrightarrow \quad \sum_{i=1}^{n}\left(\alpha_i \oplus \beta_i\right) \le \sum_{i=1}^{n}\left(\alpha_i + \beta_i\right) = \sum_{i=1}^{n}\alpha_i + \sum_{i=1}^{n}\beta_i$$

$$\longrightarrow \quad w[\alpha \oplus \beta] \le w[\alpha] + w[\beta]$$

QED

Lemma A5.4.1: Let α, β & μ be three n-tuples with coefficients in GF(2). Then the following, called the *triangle inequality*, holds true:

$$d(\alpha,\beta) + d(\beta,\mu) \ge d(\alpha,\mu) \qquad \text{(A5.4.3)}$$

Proof: Lemma A5.4.1 follows from Theorem A5.4.1:

$$d\left(\alpha,\beta\right) + d\left(\beta,\mu\right) = w\left[\alpha \oplus \beta\right] + w\left[\beta \oplus \mu\right] \ge$$

$$w\left[(\alpha \oplus \beta) \oplus (\beta \oplus \mu)\right] = w\left[\alpha \oplus \mu\right] = d\left(\alpha,\mu\right)$$

QED

A5.4.2. Proof of Theorem 5.7

Assume that the 1st constraint-length $[r]_m$ of the received sequence r, contains no more than $t \triangleq \lfloor (d_{min}-1)/2 \rfloor$ channel errors. Then,

$$w\left[[e]_m\right] \le t \triangleq \lfloor (d_{min}-1)/2 \rfloor \qquad \text{(A)}$$

Since the channel is assumed to be an additive-noise one,

$$e = v + r \longrightarrow w\left[[e]_m\right] = w\left[[r]_m + [v]_m\right] = d\left([r]_m,[v]_m\right) \qquad \text{(B)}$$

Consider now another transmitted codeword v'. By the triangle inequality,

$$d\Big([r]_m,[v']_m\Big) \geq d\Big([v']_m,[v]_m\Big) - d\Big([v]_m,[r]_m\Big) \implies$$

$$d\Big([r]_m,[v']_m\Big) \geq d\Big([v']_m,[v]_m\Big) - w\Big[[e]_m\Big] \quad \text{[using (B)]}$$

$$\implies \quad d\Big([r]_m,[v']_m\Big) \geq d\Big([v']_m,[v]_m\Big) - t \quad \text{[using (A)]} \qquad \text{(C)}$$

Since, d_{min} is the minimum distance between any two codewords, over the 1st constraint-length (the c/ws differing only in the first source block),

$d\Big([v]_m,[v']_m\Big) \geq d_{min}$ and using inequality (C):

$$d\Big([r]_m,[v']_m\Big) \geq d_{min} - t \qquad \text{(D)}$$

From (A), if d_{min} = odd, then, $d_{min} - t = d_{min} - (d_{min}-1)/2 = (d_{min}+1)/2 > t$, while if d_{min} = even, then, $d_{min} - t = d_{min} - (d_{min}-2)/2 = 1 + d_{min}/2 > t$. So, from (D): $d\Big([r]_m,[v']_m\Big) > t$.

Hence, the distance between the received truncated sequence $[r]_m$ and the transmitted truncated codeword $[v]_m$ is smaller ($\leq t$) than the distance between $[r]_m$ and any other codeword.

So, over the 1st constraint-length, no codeword v', that differs from the transmitted codeword v over the 1st source block ($[u']_0 \neq [u]_0$), is closer to r than the transmitted codeword v, if no more than t errors occured over the first constraint-length ($\iff$ the weight of $[e]_m$ is $\leq t$).

Consider now t+1 channel errors over the 1st constraint-length. Then relns (A) & (B) are modified, as following:

$$w\Big[[e]_m\Big] = d\Big([r]_m,[v]_m\Big) = t+1 \qquad \text{(E)}$$

Since d_{min} is the minimum distance, there will be a codeword v' such that $d([v]_m,[v']_m) \geq d_{min}$, where v & v' differ only over the 1st source block. Then from (A5.4.3),

$$\implies \quad d\Big([r]_m,[v']_m\Big) \geq d_{min} - t - 1 \qquad \text{(F)}$$

If d_{min} = odd, then, $d_{min} - t - 1 = d_{min} - (d_{min}-1)/2 - 1 = (d_{min}-1)/2 = t$. If d_{min} = even, then, $d_{min} - t - 1 = d_{min} - (d_{min}-2)/2 - 1 = (d_{min}-2) - (d_{min}-2)/2 + 1 = t + 1$. Hence, from relns (E) & (F), one concludes that there exists at least one codeword which is as close to the received sequence as is the transmitted codeword $[v]_m$ (or closer, if d_{min} = odd).

QED

A5.4.3. Proof of Theorem 5.8

Majority-logic decoding is based on the 1st constraint-length. If at least J parity checks can be formed on the error bits of the 1st block, then this decoder can correct $\lfloor J/2 \rfloor$ or fewer errors in the 1st constraint-length. Obviously, this capability cannot exceed t:

$$\lfloor J/2 \rfloor \le \lfloor (d_{min}-1)/2 \rfloor \quad \Longrightarrow$$

$$\Longrightarrow \begin{cases} \lfloor (d_{min}-1)/2 \rfloor + (-J/2) \ge 0 & /J=\text{even} \\ \lfloor (d_{min}-1)/2 \rfloor + [-(J-1)/2] \ge 0 & /J=\text{odd} \end{cases}$$

$$\Longrightarrow \begin{cases} \lfloor (d_{min}-1)/2 - J/2 \rfloor \ge 0 & /J=\text{even} \\ \lfloor (d_{min}-1)/2 - (J-1)/2 \rfloor \ge 0 & /J=\text{odd} \end{cases}$$

$$\Longrightarrow \left.\begin{cases} d_{min}-1-J \ge 0 & /J=\text{even} \\ d_{min}-J \ge 0 & /J=\text{odd} \end{cases}\right\} \Longrightarrow J \le d_{min}-1 \qquad \text{(A5.4.4)}$$

QED

APPENDIX 5.5: PROOF OF THEOREM 5.9

i) The syndrome bits that check on $e_h^{(i)}$ are orthogonal on $e_h^{(i)}$ /i=1,2,...,k & h≥0. Then this will be true for h=0, hence according to Definition 5.4*, the code is a CSOC.

ii) The code is a CSOC. Assume that there exists an $\alpha \ge 0$ for which, the syndrome bits that check on $e_\alpha^{(\mu)}$ $(1 \le \mu \le k)$ are not orthogonal on $e_\alpha^{(\mu)}$. Then, there will exist two syndrome bits, say, $s_\tau^{(\sigma)}$ & $s_{\tau'}^{(\sigma')}$ that will check on $e_\alpha^{(\mu)}$ and on some other error bit, say, $e_{\alpha'}^{(\mu')}$ (where, either $\alpha \ne \alpha'$, or $\mu \ne \mu'$, or both). Consequently:

$$s_\tau^{(\sigma)} = e_\alpha^{(\mu)} + e_{\alpha'}^{(\mu')} + \cdots \text{ [sum of other error bits]} \qquad \text{(A)}$$

$$s_{\tau'}^{(\sigma')} = e_\alpha^{(\mu)} + e_{\alpha'}^{(\mu')} + \cdots \text{ [sum of other error bits]} \qquad \text{(B)}$$

Comparing with the general syndrome eqn [(5.7), p. 138],

* See p. 138.

one can deduce the following results:

From eqn (A): $$g^{(\mu)}_{k+\sigma,\tau-a} = g^{(\mu')}_{k+\sigma,\tau-a'} = 1 \qquad \text{(C)}$$

From eqn (B): $$g^{(\mu)}_{k+\sigma',\tau'-a} = g^{(\mu')}_{k+\sigma',\tau'-a'} = 1 \qquad \text{(D)}$$

where [again from eqn (5.7)],

$$\left.\begin{array}{c} 1\le\mu,\mu'\le k,\ 1\le\sigma,\sigma'\le n-k \ \text{ and} \\ 0\le\tau-a,\tau-a'\le\text{MIN}\{\tau,m\}\ \ \&\ \ 0\le\tau'-a,\tau'-a'\le\text{MIN}\{\tau',m\} \end{array}\right] \longrightarrow \qquad \text{(E)}$$

Without loss of generality, one may assume that:

$$a'\ge a \qquad \text{(F)}$$

Consider now syndrome bits $s^{(\sigma)}_{\tau-a}$ & $s^{(\sigma')}_{\tau'-a}$. From the general syndrome eqn [(5.7)],

$$s^{(\sigma)}_{\tau-a} = e^{(k+\sigma)}_{\tau-a} + \sum_{z=0}^{\theta}\sum_{i=1}^{k} e^{(i)}_{\tau-a-z}\, g^{(i)}_{k+\sigma,z} \qquad \text{(A5.5.1)}$$

where: $\theta \triangleq \text{MIN}\{m,\tau-a\}$.

$$s^{(\sigma')}_{\tau'-a} = e^{(k+\sigma')}_{\tau'-a} + \sum_{z=0}^{\theta'}\sum_{i=1}^{k} e^{(i)}_{\tau'-a-z}\, g^{(i)}_{k+\sigma',z} \qquad \text{(A5.5.2)}$$

where: $\theta' \triangleq \text{MIN}\{m,\tau'-a\}$.

Consider eqn (A5.5.1); from (E), $0 \le \tau-a \le \text{MIN}\{\tau,m\}$. So $\tau-a \le m$ and hence, $\theta = \tau-a$. Then $z = \tau-a$ is a permitted value because $0 \le z \le \theta$. Also, from (F), $a' \ge a \Longleftrightarrow \tau-a' \le \tau-a$. Since, from (E) $0 \le \tau-a'$, $z = \tau-a'$ is also a permitted value. With respect to eqn (A5.5.1), consider the two terms defined by $z = \tau-a$ & $i = \mu$ and $z = \tau-a'$ & $i = \mu'$:

$$s^{(\sigma)}_{\tau-a} = \Big[e^{(i)}_{\tau-a-z}g^{(i)}_{k+\sigma,z}\Big]^{i=\mu}_{z=\tau-a} + \Big[e^{(i)}_{\tau-a-z}g^{(i)}_{k+\sigma,z}\Big]^{i=\mu'}_{z=\tau-a'} + \Big[\ \cdots\ \Big] \longrightarrow$$

$$\longrightarrow\ s^{(\sigma)}_{\tau-a} = \Big[e^{(\mu)}_{0}g^{(\mu)}_{k+\sigma,\tau-a}\Big] + \Big[e^{(\mu')}_{a'-a}g^{(\mu')}_{k+\sigma,\tau-a'}\Big] + \Big[\ \cdots\ \Big] \qquad \text{(G)}$$

Similarly, from eqn (A5.5.2), using the same arguments as above, $z = \tau'-a$ & $z = \tau'-a'$ are permitted values of z. Consider the terms $z = \tau'-a$ & $i = \mu$ and $z = \tau'-a'$ & $i = \mu'$:

$$s^{(\sigma')}_{\tau'-a} = \Big[e^{(i)}_{\tau'-a-z}g^{(i)}_{k+\sigma',z}\Big]^{i=\mu}_{z=\tau'-a} + \Big[e^{(i)}_{\tau'-a-z}g^{(i)}_{k+\sigma',z}\Big]^{i=\mu'}_{z=\tau'-a'} + \Big[\ \cdots\ \Big] \longrightarrow$$

$$\longrightarrow\ s^{(\sigma')}_{\tau'-a} = \Big[e^{(\mu)}_{0}g^{(\mu)}_{k+\sigma',\tau'-a}\Big] + \Big[e^{(\mu')}_{a'-a}g^{(\mu')}_{k+\sigma',\tau'-a'}\Big] + \Big[\ \cdots\ \Big] \qquad \text{(H)}$$

Note that the four g-coefficients shown explicitly in eqns (G) & (H) are all equal to 1 [according to eqns (C) & (D)]. Then,

$$s^{(\sigma)}_{\tau-a} = e^{(\mu)}_{0} + e^{(\mu')}_{a'-a} + \left[\cdots \right] \qquad (I)$$

$$s^{(\sigma')}_{\tau'-a} = e^{(\mu)}_{0} + e^{(\mu')}_{a'-a} + \left[\cdots \right] \qquad (J)$$

Syndrome bits $s^{(\sigma)}_{\tau-a}$ & $s^{(\sigma')}_{\tau'-a}$ both check on $e^{(\mu)}_{0}$, but they also check on $e^{(\mu')}_{a'-a}$. Since, by hypothesis, $a'-a \neq 0$, or $\mu \neq \mu'$, the code is not a CSOC which <u>contradicts</u> the initial hypothesis, hence the assumption was not correct, hence the syndrome bits that check on $e^{(\mu)}_{a}$ are orthogonal on $e^{(\mu)}_{a}$, for all $a \geq 0$.

QED

APPENDIX 5.6: <u>PARITY-TRIANGLES & PARITY-SQUARES FOR CSOC</u>

The fact that a code which is self-orthogonal for FD is also self-orthogonal for DD, may be exploited to limit the discussion to parity-triangles.

For a code to be a CSOC, all syndromes that check error bit $e^{(i)}_{0}$ $(1 \leq i \leq k)$ must be orthogonal on $e^{(i)}_{0}$. This means that apart from $e^{(i)}_{0}$, these syndromes should not check any other error bit twice.

According to Theorem 5.6 [see reln (5.16)], if $0 \leq a \leq m$, $1 \leq \mu \leq k$, $0 \leq \tau \leq m$ & $1 \leq \sigma \leq n-k$, then $s^{(\sigma)}_{\tau}$ checks on $e^{(\mu)}_{a}$ iff $g^{(\mu)}_{k+\sigma,\tau-a} = 1$. Then, the syndromes that check on $e^{(i)}_{0}$ $/1 \leq i \leq k$, correspond to 1s along the first column of parity triangles Δ^{j}_{i} $/j=1,2,\ldots,n-k$. If the triangle matrix, $H(\Delta)$, is considered and, say, an arrow indicates the rows that contain 1s along the 1st column of these triangles, then these arrows indicate, in effect, the syndrome bits that check on $e^{(i)}_{0}$. Any 1s along the 'arrowed' rows, apart from 1s in the 1st column, indicate other error bits that are checked by the corresponding syndrome and are 'marked' by, say, replacing them with ■. According to Definitions 5.4 & 5.2, no other error bit should be checked twice, hence no two ■s should appear along the same column. If this is the case, the code is self-orthogonal and this has to be the case if the code is to be a self-orthogonal one.

The following examples will help clarify these ideas.

Example A5.6.1: Consider the (2,1,6) systematic code with generator polynomial $g_2^{(1)}(D) = 1+D^2+D^5+D^6$, already examined in Examples A5.2.1 & A5.3.1. Since k = 1, there is only one bit in e_0. From the previous analysis its (only) parity triangle, complete with its arrows and ■s, appears below:

```
——>  ■
     0 1
——>  ■ 0 ■
     0 1 0 1
     0 0 1 0 1
——>  ■ 0 0 ■ 0 ■
——>  ■ ■ 0 0 ■ 0 ■
```

where the position of the error bit on which the syndromes (whose position is arrowed) are orthogonal, has been highlighted.

Note that each column contains no more than one ■, hence all J = 4 syndromes checking on $e_0^{(1)}$ are orthogonal on this bit. The size of the four syndromes is 1,2,3 & 4, hence the effective constraint-length, n_E, is 1+(1+2+3+4) = 11. Hence, this code will correctly estimate the error bit $e_0^{(1)}$ whenever $\lfloor J/2 \rfloor = 2$ or fewer errors occur among the 11 bits of the effective constraint-length, which are confined of course within one actual constraint-length $n_A = (m+1)n = 14$.

Although it is not necessary, one may repeat the above for the parity square. According to Theorem 5.5, the syndromes checking on $e_a^{(\mu)}$, correspond to 1s along the (m+1)th column of parity squares Γ_μ^j /j=1,2,...,n-k. From Example A5.2.1:

```
——> ■ ■ 0 0 ■ 0 ■
      1 1 0 0 1 0 1
——>     ■ ■ 0 0 ■ 0 ■
          1 1 0 0 1 0 1
            1 1 0 0 1 0 1
——>           ■ ■ 0 0 ■ 0 ■
——>             ■ ■ 0 0 ■ 0 ■
```

where highlighted is the position of the error bit on which the syndromes are orthogonal.

The above arrangement shows that the code is self-orthogonal under DD (as expected), and it also, graphically, illustrates, the reason.

∎

Example A5.6.2: Consider the (3,2,13) systematic convolutional code with generator polynomials $g_3^{(1)} = 1+D^8+D^9+D^{12}$ & $g_3^{(2)} = 1+D^6+D^{11}+D^{13}$, examined also in Example A5.3.2. Since k = 2, there are two arrangements of the parity triangles to be considered, one for each of $e_0^{(1)}$ & $e_0^{(2)}$:

The arrangement for $e_0^{(1)}$:

```
—>■                            ■
   0 1                         0 1
   0 0 1                       0 0 1
   0 0 0 1                     0 0 0 1
   0 0 0 0 1                   0 0 0 0 1
   0 0 0 0 0 1                 0 0 0 0 0 1
   0 0 0 0 0 0 1               1 0 0 0 0 0 1
   0 0 0 0 0 0 0 1             0 1 0 0 0 0 0 1
—>■ 0 0 0 0 0 0 0 ■            0 0 ■ 0 0 0 0 0 ■
—>■ ■ 0 0 0 0 0 0 0 ■          0 0 0 ■ 0 0 0 0 0 ■
   0 1 1 0 0 0 0 0 0 0 1       0 0 0 0 1 0 0 0 0 0 1
   0 0 1 1 0 0 0 0 0 0 0 1     1 0 0 0 0 1 0 0 0 0 0 1
—>■ 0 0 ■ ■ 0 0 0 0 0 0 0 ■    0 ■ 0 0 0 0 ■ 0 0 0 0 0 ■
   0 1 0 0 1 1 0 0 0 0 0 0 0 1 1 0 1 0 0 0 0 1 0 0 0 0 0 1
```

The arrangement for $e_0^{(2)}$:

```
—>■                            ■
   0 1                         0 1
   0 0 1                       0 0 1
   0 0 0 1                     0 0 0 1
   0 0 0 0 1                   0 0 0 0 1
   0 0 0 0 0 1                 0 0 0 0 0 1
—>0 0 0 0 0 0 ■                ■ 0 0 0 0 0 ■
   0 0 0 0 0 0 0 1             0 1 0 0 0 0 0 1
   1 0 0 0 0 0 0 0 1           0 0 1 0 0 0 0 0 1
   1 1 0 0 0 0 0 0 0 1         0 0 0 1 0 0 0 0 0 1
   0 1 1 0 0 0 0 0 0 0 1       0 0 0 0 1 0 0 0 0 0 1
—>0 0 ■ ■ 0 0 0 0 0 0 0 ■      ■ 0 0 0 0 ■ 0 0 0 0 0 ■
   1 0 0 1 1 0 0 0 0 0 0 0 1   0 1 0 0 0 1 0 0 0 0 0 0 1
—>0 ■ 0 0 ■ ■ 0 0 0 0 0 0 0 ■ ■ 0 ■ 0 0 0 0 ■ 0 0 0 0 0 ■
```

Note that each of $e_0^{(1)}$ and $e_0^{(2)}$ is checked by J=4 syndromes; furthermore, these sets of four syndromes are orthogonal on their corresponding error bits, because no two ■s can be found along any of the columns of the triangles. Hence the code is a CSOC and can correct 2 or fewer errors from among the n_E error bits of its effective constraint-length.

The actual constraint-length is $n_A = (m+1)n = 14 \times 3 = 42$.

Note that n_E, for the decoding of a certain error bit, equals the sum of the sizes of all syndromes checking on that bit plus one (see Definition 5.5). From the triangles above, the size of each syndrome equals the number of ■s along the 'arrowed' row plus 1 (for the corresponding parity error bit). So for each of the two cases, n_E should equal the total number of arrows, which is 4 (that accounts for the parity error bits) plus the total number of ■s, which is 14 for $e_0^{(1)}$ and 15 for $e_0^{(2)}$ (that accounts for the total number of message error bits) plus one (that accounts for the error bit checked by the set). Then the effective constraint-length for $e_0^{(1)}$ is 19 and for $e_0^{(2)}$ is 20.

■

APPENDIX 5.7: <u>BLOCK EFFECTIVE CONSTRAINT-LENGTH FOR A CSOC</u>

With Definition 5.9 in mind, let us now try to determine the N_E of a CSOC, under FD. Note from the general syndrome eqn (A5.3.4a) that each column of H(◺) corresponds to a specific information error bit; for example, a 'one' anywhere along the first column of H(◺), implies that $e_0^{(1)}$ is checked by a syndrome bit (in fact by the syndrome bit of the row in which this 'one' appears). In general, according to Theorem 5.6, for $\mu\in[1,k]$, $\alpha\in[0,m]$, $\sigma\in[1,n-k]$ & $\tau\in[0,m]$ syndrome bit $s_\tau^{(\sigma)}$ checks on error bit $e_\alpha^{(\mu)}$, iff $g_{k+\sigma,\tau-\alpha}^{(\mu)} = 1$, i.e. if the element of H(◺) in the $[(\mu-1)(m+1)+\alpha+1]$th column & $[(\sigma-1)(m+1)+\tau+1]$th row, of H(◺), is 1. Also, by inspection of eqn (A5.3.4a), syndrome bit $s_\tau^{(\sigma)}$ checks on parity error bit $e_\tau^{(k+\sigma)}$; hence one parity error bit must be considered, iff the corresponding syndrome bit is a member of an orthogonal set. Taking into account the above conclusions, the following set of instructions may be proposed for the calculation of the N_E of a CSOC.

<u>Note A5.7.1</u>: Let an (n,k,m) systematic CSOC, with triangle matrix H(◺). To calculate the block effective constraint-length of this code, under FD:

i) Inspect column $[(\mu-1)(m+1)+1]$ /$\mu=1,2,\ldots,k$ of H(◺); the 1s indicate the syndromes orthogonal on $e_0^{(\mu)}$. For $\mu=1,2,.$

..,k, let the set of ordered pairs $R_\mu \triangleq \{(\sigma,\tau)$ /$1 \le \sigma \le n-k$ & $0 \le \tau \le m$: position $[(\sigma-1)(m+1)+\tau+1]$ of column $[(\mu-1)(m+1)+1]$ is 'one'$\}$. By (A5.3.5):

$$R_\mu = \left\{(\sigma,\tau) \ : \ g^{(\mu)}_{k+\sigma,\tau} = 1 \quad /1 \le \sigma \le n-k \ \& \ 0 \le \tau \le m\right\} \quad /1 \le \mu \le k \qquad \text{(A5.7.1)}$$

Let $|R_\mu|$, the number of elements of R_μ, be J_μ. J_μ is the number of syndromes orthogonal on $e_0^{(\mu)}$ and R_μ indicates both the set of syndromes orthogonal on $e_0^{(\mu)}$ and the set of parity error bits that are checked by this set of syndromes. In other words, R_μ is concerned with the 1st column of the μth column of parity triangles (i.e. with the 1st column of Δ_μ^j /$j=1,2,\ldots,n-k$). Specifically, $(\sigma,\tau) \in R_\mu$ if there is a 'one' in Δ_μ^σ, in column 1 and row τ, where τ ranges from 0 to m. Conversely, if $(\sigma,\tau) \in R_\mu$, then error bit $e_0^{(\mu)}$ is checked by syndrome bit $s_\tau^{(\sigma)}$.

ii) Let the union of the R_μs: $R \triangleq R_1 \cup R_2 \cup \cdots \cup R_k$. R indicates both the complete set of distinct syndromes that check on the 1st information error block $\mathbf{e}_0$ and the set of distinct parity error bits that are checked by this set of syndromes. Hence, $|R|$ is the contribution of the parity error bits towards N_E. R is the set of rows of $H(\Delta)$ that are 'involved' in the estimation of $\mathbf{e}_0$. From (A5.7.1):

$$R = \left\{(\sigma,\tau) \ : \ g^{(\mu)}_{k+\sigma,\tau} = 1 \quad /1 \le \mu \le k, \ 1 \le \sigma \le n-k \ \& \ 0 \le \tau \le m\right\} \qquad \text{(A5.7.2)}$$

iii) Consider now the set, C, of message error bits, exclusive of $\mathbf{e}_0$, that are checked by the set of syndromes indicated by R. Inspect all columns of $H(\Delta)$, apart from columns $[(\sigma-1)(m+1)+1]$ (i.e. apart from the 1st column of each parity triangle), for at least one 'one' in a position (row) specified by R. A row of $H(\Delta)$ can be written as (σ,τ), where σ is the row of triangles ($1 \le \sigma \le n-k$) and τ is the row within the triangle ($0 \le \tau \le m$). Similarly, a column may be written as (β,α) ($1 \le \beta \le k$ & $0 \le \alpha \le m$). Then, C may be defined as the set of those columns (β,α) which contain a 'one' in at least one of the rows of R: $C \triangleq \{(\beta,\alpha)$ /$1 \le \beta \le k$ & $0 \le \alpha \le m$: element $[(\sigma,\tau),(\beta,\alpha)]$ of $H(\Delta)$ is 'one', for all $(\sigma,\tau) \in R\}$. From the discussion in Appendix 5.3:

$$C = \left\{(\beta,\alpha)\ /1\le\beta\le k\ \&\ 0\le\alpha\le m\ :\ g^{(\beta)}_{k+\sigma,\tau-\alpha} = 1\ \&\ (\sigma,\tau)\in R\right\} \qquad (A5.7.3)$$

Then, $|C|$ is the contribution of the message error bits, exclusive of the ones that are checked (i.e. of e_0), towards N_E. C is the set of columns of H(ɮ) that 'participate' in the estimation of e_0.

iv) Then: $$N_E = k + |R| + |C| \qquad (A5.7.4)$$

■

Example A5.7.1: Consider now the calculation of the block effective constraint-length (under FD), N_E, for the (3,2,13) systematic code with generator polynomials $g_3^{(1)} = 1+$ $+D^8+D^9+D^{12}$ and $g_3^{(2)} = 1+D^6+D^{11}+D^{13}$, examined also in Examples A5.3.2. & A5.6.2.

From above:

$$g^{(1)}_{3,0} = g^{(1)}_{3,8} = g^{(1)}_{3,9} = g^{(1)}_{3,12} = g^{(2)}_{3,0} = g^{(2)}_{3,6} = g^{(2)}_{3,11} = g^{(2)}_{3,13} = 1$$

Using the instructions of Note A5.7.1:

$$R = \left\{(\sigma,\tau)\ :\ g^{(\mu)}_{k+\sigma,\tau} = 1\quad /1\le\mu\le2,\ 1\le\sigma\le1\ \&\ 0\le\tau\le13\right\} \longrightarrow$$

$$R = \left\{(1,\tau)\ :\ g^{(\mu)}_{3,\tau} = 1\quad /\mu=1,2\ \&\ 0\le\tau\le13\right\} \longrightarrow$$

$$R = \left\{(1,\tau)\ /\tau=0,6,8,9,11,12,13\right\} \longrightarrow \quad |R| = 7$$

Hence, the rows of H(ɮ) to be examined, are along the 1st row of triangles, and specifically rows 0, 6, 8, 9, 11, 12 & 13. Then, excluding columns (1,0) & (2,0) (i.e. the 1st column of each of the two parity triangles), columns (1,x) /x=1,2,3,4,5,6,8,9,11,12,13 & (2,y) /y=1,2,3,5,6,7,8,9,11, 12,13 contain a 'one' along at least one of the rows of R (see pp. 396 & 404). Hence, $|C| = 22$. Alternatively:

$$C = \left\{(\beta,\alpha)\ /1\le\beta\le2\ \&\ 1\le\alpha\le13\ :\ g^{(\beta)}_{3,\tau-\alpha} = 1\ \&\ (1,\tau)\in R\right\} \longrightarrow$$

$$C = \left\{(\beta,\alpha)\ /1\le\beta\le2\ \&\ 1\le\alpha\le13\ :\ g^{(\beta)}_{3,\tau-\alpha} = 1\ \&\ \tau=0,6,8,9,11,12,13\right\}$$

$$\longrightarrow \quad C = \left\{(1,\alpha)\ /1\le\alpha\le13\ :\ g^{(1)}_{3,\tau-\alpha} = 1\ \&\ \tau=0,6,8,9,11,12,13\right\}\ \cup$$

$$\cup\ \left\{(2,\alpha)\ /1\le\alpha\le13\ :\ g^{(2)}_{3,\tau-\alpha} = 1\ \&\ \tau=0,6,8,9,11,12,13\right\} \longrightarrow$$

$$C = \left\{(1,\alpha)\ /1\le\alpha\le13\ :\ \tau-\alpha = 0,8,9,12\ \&\ \tau=0,6,8,9,11,12,13\right\}\ \cup$$

$$\cup\ \left\{(2,\alpha)\ /1\le\alpha\le13\ :\ \tau-\alpha = 0,6,11,13\ \&\ \tau=0,6,8,9,11,12,13\right\}$$

$$\Longrightarrow$$

$$C = \left\{(1,\alpha)\ /\alpha\geq 1\ :\ \alpha=\tau,\tau-8,\tau-9,\tau-12\ \&\ \tau=0,6,8,9,11,12,13\right\}\ U$$

$$U\ \left\{(2,\alpha)\ /\alpha\geq 1\ :\ \alpha=\tau,\tau-6,\tau-11,\tau-13\ \&\ \tau=0,6,8,9,11,12,13\right\}$$

$$\Longrightarrow$$

$$C = \left\{(1,\alpha)\ /\alpha=6,8,9,11,12,13,1,3,4,5,2,3,4,1\right\}\ U$$

$$U\ \left\{(2,\alpha)\ /\alpha=6,8,9,11,12,13,2,3,5,6,7,1,2\right\}$$

$$\Longrightarrow$$

$$C = \left\{(1,\alpha)\ /\alpha=1,2,3,4,5,6,8,9,11,12,13\right\}\ U$$

$$U\ \left\{(2,\alpha)\ /\alpha=1,2,3,5,6,7,8,9,11,12,13\right\}$$

$$\Longrightarrow \quad |C| = 11+11$$

Then, from (A5.7.4): $N_E = 2 + 7 + 22 = 31$

In order to verify the above results consider the syndromes checking on block e_0. From Example A5.3.2 (p. 395):

$$s_0^{(1)} = e_0^{(1)} + e_0^{(2)} + e_0^{(3)}$$

$$s_8^{(1)} = e_0^{(1)} + e_8^{(1)} + e_2^{(2)} + e_8^{(2)} + e_8^{(3)}$$

$$s_9^{(1)} = e_0^{(1)} + e_1^{(1)} + e_9^{(1)} + e_3^{(2)} + e_9^{(2)} + e_9^{(3)}$$

$$s_{12}^{(1)} = e_0^{(1)} + e_3^{(1)} + e_4^{(1)} + e_{12}^{(1)} + e_1^{(2)} + e_6^{(2)} + e_{12}^{(2)} + e_{12}^{(3)}$$

$$s_0^{(1)} = e_0^{(1)} + e_0^{(2)} + e_0^{(3)} \ *$$

$$s_6^{(1)} = e_6^{(1)} + e_0^{(2)} + e_6^{(2)} + e_6^{(3)}$$

$$s_{11}^{(1)} = e_2^{(1)} + e_3^{(1)} + e_{11}^{(1)} + e_0^{(2)} + e_5^{(2)} + e_{11}^{(2)} + e_{11}^{(3)}$$

$$s_{13}^{(1)} = e_1^{(1)} + e_4^{(1)} + e_5^{(1)} + e_{13}^{(1)} + e_0^{(2)} + e_2^{(2)} + e_7^{(2)} + e_{13}^{(2)} + e_{13}^{(3)}$$

The above equations will now verify the predictions. The syndrome bits that check e_0, check 19+20 = 39 error bits, while N_E was found to be 31. Then, 39-31 = 8 error bits must be found to be duplicated, in the above equations. Furthermore, the actual constraint-length is 42, hence 42-31 = 11 bits of the actual constraint-length must be found to be missing from the above eqns. The error bits that participate in the eqns above are:

* This eqn is identical with the 1st one, but is repeated for symmetry.

$e_a^{(1)}$ /a = 0,8,1,9,3,4,12,0,6,2,3,11,1,4,5,13.

$e_a^{(2)}$ /a = 0,2,8,3,9,1,6,12,0,6,5,11,2,7,13.

$e_a^{(3)}$ /a = 0,8,9,12,0,6,11,13.

From $e_a^{(1)}$, a = 0,1,3,4 are *duplicated* and a = 7,10 are *missing*.

From $e_a^{(2)}$, a = 0,2,6 are *duplicated* and a = 4,10 are *missing*.

From $e_a^{(3)}$, a = 0 is *duplicated* and a = 1,2,3,4,5,7,10 are *missing*.

Then, indeed 8 bits are *duplicated* and 11 are *missing*.

∎

APPENDIX 5.8: DISTANCE PROPERTIES OF CSOCs

A5.8.1. Proof of Theorem 5.11

According to eqns (A2.5.4) & (A2.5.5) (pp. 310-1),

$$d_{min} \triangleq \mathrm{MIN}\left\{w[v]_m : \quad [u]_0 \neq 0\right\} \qquad \text{(A5.8.1)}$$

Recall from eqn (2.67) that, if v is a codeword (c/w), then $vH^T = 0$. Consider $[v]_m$, the 1st constraint-length of v; this is a 1 x n(m+1) row vector. Consider, also, a suitable truncation of H; from Definition 2.13, $[H]_m$ is an (m+1) x (m+1) matrix of (n-k) x n submatrices, i.e. $[H]_m^T$ is an (m+1)n x (m+1)(n-k) matrix. So:

$$\text{If } v \text{ is a c/w, then } \left[v\right]_m\left[H\right]_m^T = 0 \qquad \text{(A5.8.2)}$$

where 0 is the 1 x (n-k)(m+1) zero row-vector.

Note from eqn (A5.8.2) that the matrix product equals the sum of the rows of $[H]_m^T$ that correspond to 'ones' in $[v]_m$. Since this product is zero, any sum of rows of $[H]_m^T$ or, the same, any sum of columns of $[H]_m$ that is zero corresponds to a codeword. Furthermore, one may require that this c/w is such that its 1st information block $[u]_0$ is non-zero; this restriction is imposed so that one will be able to calculate d_{min} [see eqn (A5.8.1)]. The restriction that $[u]_0 \neq 0$ is equivalent to $[v^m]_0 \neq 0$, since $v^m = u$, for systematic codes. Hence, any sum of columns of $[H]_m$, including at least one of the first k, that is zero, corresponds to a c/w which is non-zero in its first information block.

QED

A5.8.2. Proof of Theorem 5.12

From the general form of H for an (n,k,m) systematic convolutional code (see Theorem 2.11), and the directions for the construction of $[H]_m$ (see Theorem 5.11):

$$[H]_m = \begin{bmatrix} P_0^T & I & & & & & & & \\ P_1^T & 0 & P_0^T & I & & & & & \\ P_2^T & 0 & P_1^T & 0 & P_0^T & I & & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & & & \\ P_m^T & 0 & P_{m-1}^T & 0 & P_{m-2}^T & 0 & \cdots & P_0^T & I \end{bmatrix} \qquad (A5.8.3)$$

According to Theorem 5.11, to calculate d_{min} one would have to consider at least one of the first k columns, i.e. at least one of the columns of $[P_0\, P_1 \cdots P_m]^T$.

According to Theorem 2.6:

$$\text{For all } z=0,1,..,m: \quad P_z = \begin{bmatrix} g^{(1)}_{k+1,z} & g^{(1)}_{k+2,z} & \cdots & g^{(1)}_{n,z} \\ g^{(2)}_{k+1,z} & g^{(2)}_{k+2,z} & \cdots & g^{(2)}_{n,z} \\ \vdots & \vdots & & \vdots \\ g^{(k)}_{k+1,z} & g^{(k)}_{k+2,z} & \cdots & g^{(k)}_{n,z} \end{bmatrix} \qquad (A5.8.4)$$

From eqns (A5.8.3) & (A5.8.4) C_μ, the μth column of $[H]_m$, contains the following elements ($1 \le \mu \le k$):

$$C_\mu \triangleq \left(g^{(\mu)}_{k+1,0}\, g^{(\mu)}_{k+2,0} \cdots g^{(\mu)}_{n,0}\, g^{(\mu)}_{k+1,1}\, g^{(\mu)}_{k+2,1} \cdots g^{(\mu)}_{n,1} \cdots g^{(\mu)}_{k+1,m}\, g^{(\mu)}_{k+2,m} \cdots g^{(\mu)}_{n,m}\right)$$

If the weight of C_μ is considered, a rearrangement of the elements is permissible:

$$w\left[C_\mu\right] = w\left[g^{(\mu)}_{k+1,0}\, g^{(\mu)}_{k+1,1}\cdots g^{(\mu)}_{k+1,m}\, g^{(\mu)}_{k+2,0}\, g^{(\mu)}_{k+2,1}\cdots g^{(\mu)}_{k+2,m}\cdots g^{(\mu)}_{n,0}\, g^{(\mu)}_{n,1}\cdots g^{(\mu)}_{n,m}\right]$$

$$\Longrightarrow \quad w\left[C_\mu\right] = w\left[g^{(\mu)}_{k+1}, g^{(\mu)}_{k+2}, \ldots, g^{(\mu)}_{n}\right]$$

$$\Longrightarrow \quad w\left[C_\mu\right] = \sum_{j=1}^{n-k} w\left[g^{(\mu)}_{k+j}\right] \quad /\mu=1,2,\ldots,k \qquad \text{(A5.8.5)}$$

The theorem follows from eqn (A5.8.5) and Theorem 5.6. Note, also, that H & $[H]_m$ are identical in their first n columns [compare eqns (2.59) & (A5.8.3)].

QED

A5.8.3. Proof of Theorem 5.13

Let any of the first k columns of $[H]_m$, say C_μ. According to Theorem 5.12, its weight is J_μ. There are always J_μ more columns of $[H]_m$ that together with C_μ sum-up to zero. This can be seen from eqn (A5.8.3). Columns k+1,...,n,n+k+1,.. .,2n,2n+k+1,..,3n,...,hn+k+1,...,(h+1)n,...,mn+k+1,..,(m+1)n correspond to the elements of the identity matrix I_{n-k}, hence they contain exactly one 'one'. Furthermore, from (A5.8.3), there are exactly (m+1)(n-k) such columns, each one with its 'one' in a different row and there are (m+1)(n-k) rows in $[H]_m$. Hence, $[H]_m$ has always J_μ columns which sum up to zero together with C_μ, hence d_{min} cannot be greater than $J_\mu+1$. In fact, since $J = \mathrm{MIN}\{J_\mu \,/\mu=1,2,\ldots,k\}$, $d_{min} \le J+1$.

QED

APPENDIX 6.1: PROOF OF THE THEORY IN SECTION 6.1

A6.1.1. Proof of Lemma 6.1

Let $\mathcal{E}_i$ /i=1,2,...,J be the bits orthogonal on e_m. Then $\mathcal{E}_i+e_m$ /1≤i≤J depends entirely on error bits other than e_m, because each $\mathcal{E}_i$ /i=1,2,...,J checks on e_m. Let $P(\Sigma=\mu|e_m=0)$ = P. Then, P is the probability that exactly μ, of the $\{\mathcal{E}_i\}$, are 1, given that $e_m=0$ or, the same, P is the probability that exactly μ of the $\{\mathcal{E}_i+e_m\}$ are 1, where $e_m=0$ or, the same, P is the probability that exactly J-μ of the $\{\mathcal{E}_i+e_m\}$ are 0, where $e_m=0$ or, the same, P is the probability that exactly J-μ of the $\{\mathcal{E}_i\}$ are 0, where $e_m=0$ or, the same, P is the probability that exactly J-μ of the $\{\mathcal{E}_i\}$ are 1, given that $e_m=1$. Hence:

$$P\left(\Sigma=\mu|e_m=0\right) = P = P\left(\Sigma=J-\mu|e_m=1\right)$$

QED

A6.1.2. Proof of Theorem 6.1

$$\delta P_d(T) < 0 \quad /T<X \quad \Longleftrightarrow \quad P_d(T-1) > P_d(T) \quad /T<X \qquad \text{(A)}$$

If X=integer, by hypothesis:

$$\delta P_d(X) = \delta P_d(\lfloor X\rfloor) = 0 \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad P_d(X) = P_d(X-1) = P_d(\lfloor X\rfloor) = P_d(\lfloor X\rfloor-1) \qquad \text{(B)}$$

If X=integer, because $\lfloor X\rfloor = X$:

$$T < X \quad \Longleftrightarrow \quad T = 1,2,\ldots,\lfloor X\rfloor-1 \qquad \text{(C)}$$

From (A), (B) & (C):

If X=integer: $\delta P_d(T) < 0 \quad /T<X \quad \& \quad \delta P_d(X) = 0 \quad \Longleftrightarrow$

$$\Longleftrightarrow \quad P_d(0) > P_d(1) > \cdots > P_d(\lfloor X\rfloor-1) = P_d(\lfloor X\rfloor) \qquad \text{(D)}$$

If X≠integer, because $\lfloor X\rfloor < X$:

$$T < X \quad \Longleftrightarrow \quad T = 1,2,\ldots,\lfloor X\rfloor-1,\lfloor X\rfloor \qquad \text{(E)}$$

From (A) & (E):

If $X \neq$ integer: $\quad \delta \mathbf{P}_d(T) < 0 \quad /T<X \quad \Longleftrightarrow$

$$\Longleftrightarrow \quad \mathbf{P}_d(0) > \mathbf{P}_d(1) > \cdots > \mathbf{P}_d(\lfloor X \rfloor -1) > \mathbf{P}_d(\lfloor X \rfloor) \qquad \text{(F)}$$

From (D) & (F):

$$\left[\delta \mathbf{P}_d(T) < 0 \quad /T<X\right] \quad \& \quad \left[\delta \mathbf{P}_d(X) = 0 \quad /X=\text{integer}\right] \quad \Longleftrightarrow$$

$$\Longleftrightarrow \quad \mathbf{P}_d(0) > \mathbf{P}_d(1) > \cdots > \mathbf{P}_d(\lfloor X \rfloor -1) \geq \mathbf{P}_d(\lfloor X \rfloor) \qquad \text{(G)}$$

Consider now the 3rd condition:

$$\delta \mathbf{P}_d(T) > 0 \quad /T>X \quad \Longleftrightarrow \quad \mathbf{P}_d(T) > \mathbf{P}_d(T-1) \quad /T>X \qquad \text{(H)}$$

$$T > X \quad \Longleftrightarrow \quad T = J, J-1, \ldots, \lfloor X \rfloor +1 > X \qquad \text{(I)}$$

From (H) & (I):

$\delta \mathbf{P}_d(T) > 0 \quad /T>X \quad \Longleftrightarrow$

$$\Longleftrightarrow \quad \mathbf{P}_d(J) > \mathbf{P}_d(J-1) > \cdots > \mathbf{P}_d(\lfloor X \rfloor +1) > \mathbf{P}_d(\lfloor X \rfloor) \qquad \text{(J)}$$

From (G) & (J):

$$\left[\delta \mathbf{P}_d(T) < 0 \quad /T<X\right] \quad \& \quad \left[\delta \mathbf{P}_d(T) > 0 \quad /T>X\right] \quad \&$$

$$\& \quad \left[\delta \mathbf{P}_d(X) = 0 \quad /X=\text{integer}\right] \quad \Longleftrightarrow \quad \underset{T}{\text{MIN}} \left\{\mathbf{P}_d(T)\right\} = \mathbf{P}_d(\lfloor X \rfloor)$$

$$\Longleftrightarrow \quad T_o = \lfloor X \rfloor$$

QED

A6.1.3. Proof of Theorem 6.2

Clearly, the mod-2 sum of c bits is 1, iff any combination of an odd number of them is 1.

Since there are $C(c,i) = c!/[i!(c-i)!]$ combinations of i things out of c, then there are C(c,i) distinct patterns of c bits of which i are 1. Because a bit assumes its value (0 or 1) independently of the other bits, then the probability that exactly i of the c bits are 1, and of course c-i are 0, is $p^i(1-p)^{c-i}$; this is the probability of one of the C(c,i) patterns mentioned above. Then,

$$P = \sum_{\substack{i=1 \\ i=\text{odd}}}^{c} \binom{c}{i} p^i (1-p)^{c-i} \qquad \text{(A)}$$

Let: $$f(i) \triangleq \begin{cases} 1 & /i=\text{odd} \\ 0 & /i=\text{even} \end{cases} \qquad \text{(B)}$$

From eqns (A) & (B):

$$P = \sum_{i=1}^{c}\binom{c}{i}f(i)p^i(1-p)^{c-i} \qquad \text{(C)}$$

$$[1-(-1)^i]/2 \quad \begin{cases} [1-(-1)]/2 = 1 & /i=\text{odd} \\ [1-1]/2 = 0 & /i=\text{even} \end{cases} \qquad \text{(D)}$$

Then, from (B) & (D): $f(i) = [1-(-1)^i]/2$ and combining with (C):

$$P = \tfrac{1}{2}\sum_{i=1}^{c}C(c,i)[1-(-1)^i]p^i(1-p)^{c-i} \Longrightarrow$$

$$P = \tfrac{1}{2}\left[\sum_{i=1}^{c}C(c,i)p^i(1-p)^{c-i} - \sum_{i=1}^{c}C(c,i)(-p)^i(1-p)^{c-i}\right]$$

$$\Longrightarrow \quad P = \tfrac{1}{2}\left\{[p+(1-p)]^c - [(-p)+(1-p)]^c\right\} = \tfrac{1}{2}[1-(1-2p)^c]$$

where the binomial expansion was used [see (A6.1.3), below].

QED

A6.1.4. Approximations to $(1-2p)^c$

Theorem A6.1.1: Let p be a small positive real number and c a positive integer. Then:

$$\lim_{p \to 0} (1-2p)^c = e^{-2pc} \qquad \text{(A6.1.1)}$$

Proof: Let

$$x \triangleq (1-2p)^c \Longleftrightarrow x = e^{c\ln(1-2p)} \qquad \text{(A)}$$

It is well known that (see Kreyszig [40], p. 579)

$$e^z = \sum_{i=0}^{+\infty} z^i/i! \qquad \text{(B)}$$

Applying (B), in (A):

$$x = \sum_{i=0}^{+\infty}[c\ln(1-2p)]^i/i! \;=\; 1 + \sum_{i=1}^{+\infty}[c\ln(1-2p)]^i/i! \;\longrightarrow$$

$$\longrightarrow \;\; \mathop{LIM}_{p\to 0}\left\{x\right\} = 1 + \mathop{LIM}_{p\to 0}\left\{ \sum_{i=1}^{+\infty}[c\ln(1-2p)]^i/i! \right\} \;\longrightarrow$$

$$\longrightarrow \;\; \mathop{LIM}_{p\to 0}\left\{x\right\} = 1 + \sum_{i=1}^{+\infty}(1/i!)\Big[c\{\mathop{LIM}_{p\to 0} \ln(1-2p)\}\Big]^i \qquad (C)$$

From Kreyszig [40], p. 580: $\mathop{LIM}_{z\to 0}\left\{\ln(1-z)\right\} = -z$ (D)

From (C) & (D):

$$\mathop{LIM}_{p\to 0}\left\{x\right\} = 1 + \sum_{i=1}^{+\infty}(1/i!)(-2pc)^i = e^{-2pc} \quad \text{[by (B)]}$$

QED

<u>Theorem A6.1.2</u>: Let p be a small positive real number and c a positive integer. Then:

$$(1-2p)^c \approx 1-2pc \quad /pc \ll 1 \qquad (A6.1.2)$$

<u>Proof</u>: It is known that (see for example Biggs [36], p. 69):

$$(\alpha+\beta)^n = \sum_{i=0}^{n}\alpha^i\beta^{n-i}\binom{n}{i} \qquad (A6.1.3)$$

From (A6.1.3):

$$(1-2p)^c = \sum_{i=0}^{c}(-2p)^i\binom{c}{i} = 1 - 2pc + \sum_{i=2}^{c}(-2p)^i\binom{c}{i} \qquad (A)$$

Consider the magnitude of the ratio of the (i+1)th, over the ith, term of the summation in the RHS of (A), where $i\in[1,c-1]$:

$$R(i) \triangleq |(-2p)^{i+1}C(c,i+1)/(-2p)^i/C(c,i)| \;\longrightarrow$$

$$R(i) = 2pc!i!(c-i)(c-i-1)!/[c!(i+1)i!(c-i-1)!] \;\longrightarrow$$

$$R(i) = 2p[(c-i)/(i+1)]$$

Since $(c-i)/(i+1)$ is a decreasing function of $i\in[1,c-1]$:

$$R(i) \le R(1) = 2p[(c-1)/(1+1)] = p(c-1) < pc$$

Hence, the terms in the summation in (A) decrease by a factor of at least pc (if pc<1), as i increases in steps of 1. For pc«1, the summation may be eliminated:

$$(1-2p)^c \approx 1-2pc \qquad /pc \ll 1$$

QED

A6.1.5. Examples of the Calculation of $P(\Sigma=\mu|e_m=0)$

Example A6.1.1: Assume that J=4 & μ=3. Then, from eqn (6.18) (p. 155):

$$P\left(\Sigma=3|e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right)\sum_{\substack{x(i)\\1\le x(i)<x(i+1)\le 4\\1\le i\le 3}} K_{x(1)}K_{x(2)}K_{x(3)} \longrightarrow$$

$$P\left(\Sigma=3|e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right)\left(K_1K_2K_3 + K_1K_2K_4 + K_1K_3K_4 + K_2K_3K_4\right) \qquad (A)$$

Alternatively, from eqn (6.19):

$$P\left(\Sigma=3|e_m=0\right) = \left(P_1P_2P_3P_4\right)\sum_{\substack{y(j)\\1\le y(j)<y(j+1)\le 4\\1\le j\le 4-3}} \left[K_{y(1)}\right]^{-1} \longrightarrow$$

$$P\left(\Sigma=3|e_m=0\right) = \left(P_1P_2P_3P_4\right)\left(K_1^{-1} + K_2^{-1} + K_3^{-1} + K_4^{-1}\right) \qquad (B)$$

∎

Example A6.1.2: Consider now some figures for the case of Example A6.1.1. Let $p=10^{-4}$ and $c_1=1$, $c_2=3$, $c_3=6$ & $c_4=12$. Then, using Theorem 6.2:

TABLE A6.1.1

i	c_i	P_i	Q_i	K_i
1	1	1.00000×10^{-4}	0.99990	1.00010×10^{-4}
2	3	2.99940×10^{-4}	0.99970	3.00030×10^{-4}
3	6	5.99700×10^{-4}	0.99940	6.00060×10^{-4}
4	12	1.19868×10^{-3}	0.99880	1.20012×10^{-3}

From TABLE A6.1.1:

$$P_1P_2P_3P_4 = 10^{-4} \times 2.9994\times10^{-4} \times 5.997\times10^{-4} \times 1.19868\times10^{-3}$$

$$\longrightarrow \quad P_1P_2P_3P_4 = 2.15708\times10^{-14} \qquad (A)$$

From eqn (B) of Example A6.1.1:

$$P\left(\Sigma=3\,|\,e_m=0\right) = (2.15708\times10^{-14})\left(9999 + 3333 + 1666.5 + 833.25\right)$$

$$\longrightarrow \qquad P\left(\Sigma=3\,|\,e_m=0\right) = 3.41504\times10^{-10}$$

∎

<u>Example A6.1.3</u>: Consider the case of the previous two examples, for $P(\Sigma=\mu|e_m=0)$ $/\mu=0,1,2,4$.

Obviously:

$$P\left(\Sigma=4\,|\,e_m=0\right) = P_1P_2P_3P_4 = 2.15708\times10^{-14} \qquad (A)$$

$$P\left(\Sigma=0\,|\,e_m=0\right) = Q_1Q_2Q_3Q_4 = 0.99780 \qquad (B)$$

From (6.18):

$$P\left(\Sigma=1\,|\,e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right) \sum_{\substack{x(i)\\ 1\le x(i)<x(i+1)\le 4\\ 1\le i\le 1}} K_{x(1)} \longrightarrow$$

$$P\left(\Sigma=1\,|\,e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right)\left(K_1+K_2+K_3K_4\right) = 2.19538\times10^{-3} \qquad (C)$$

From (6.18):

$$P\left(\Sigma=2\,|\,e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right) \sum_{\substack{x(i)\\ 1\le x(i)<x(i+1)\le 4\\ 1\le i\le 2}} K_{x(1)}K_{x(2)} \longrightarrow$$

$$P\left(\Sigma=2\,|\,e_m=0\right) = \left(Q_1Q_2Q_3Q_4\right)\left(K_1K_2+K_1K_3+K_1K_4+K_2K_3+K_2K_4+K_3K_4\right)$$

$$\longrightarrow \qquad P\left(\Sigma=2\,|\,e_m=0\right) = 1.46706\times10^{-6} \qquad (D)$$

∎

<u>Example A6.1.4</u>: Consider now the details of Example A6.1.3 and the calculation of $\mathbf{P}_d$. Since J=4, (J-1)/2 = 1.5.

From (6.23b):

$$\mathbf{P}_d(T) = \sum_{\mu=4}^{4} P\left(\Sigma=\mu\,|\,e_m=0\right) + (1-p)\sum_{\mu=1}^{3} P\left(\Sigma=\mu\,|\,e_m=0\right) \qquad /T=0 \longrightarrow$$

$$\mathbf{P}_d = P\left(\Sigma=4\,|\,e_m=0\right)+(1-p)\left[P\left(\Sigma=1\,|\,e_m=0\right)+P\left(\Sigma=2\,|\,e_m=0\right)+P\left(\Sigma=3\,|\,e_m=0\right)\right]$$

$$\longrightarrow \qquad \mathbf{P}_d = P\left(\Sigma=4\,|\,e_m=0\right)+(1-p)\left[1-P\left(\Sigma=0\,|\,e_m=0\right)-P\left(\Sigma=4\,|\,e_m=0\right)\right]$$

$$\longrightarrow \qquad \mathbf{P}_d = pP\left(\Sigma=4\,|\,e_m=0\right) + (1-p)\left[1-P\left(\Sigma=0\,|\,e_m=0\right)\right]$$

$$\longrightarrow \qquad \mathbf{P}_d(0) = 2.200\times10^{-3} \qquad (A)$$

Similarly, from (6.23b):

$$P_d(T) = \sum_{\mu=3}^{4} P\left(\Sigma=\mu \,|\, e_m=0\right) + (1-p)\sum_{\mu=2}^{2} P\left(\Sigma=\mu \,|\, e_m=0\right) \quad /T=1 \longrightarrow$$

$$\longrightarrow \quad P_d = P\left(\Sigma=3 \,|\, e_m=0\right) + P\left(\Sigma=4 \,|\, e_m=0\right) + (1-p)P\left(\Sigma=2 \,|\, e_m=0\right)$$

$$\longrightarrow \quad P_d(1) = 1.467\times10^{-6} \qquad \text{(B)}$$

From (6.23a):

$$P_d(T) = \sum_{\mu=3}^{4} P\left(\Sigma=\mu \,|\, e_m=0\right) + p\sum_{\mu=2}^{2} P\left(\Sigma=\mu \,|\, e_m=0\right) \quad /T=2 \longrightarrow$$

$$P_d = P\left(\Sigma=3 \,|\, e_m=0\right) + P\left(\Sigma=4 \,|\, e_m=0\right) + pP\left(\Sigma=2 \,|\, e_m=0\right)$$

$$\longrightarrow \quad P_d(2) = 4.882\times10^{-10} \qquad \text{(C)}$$

Similarly, from (6.23a):

$$P_d(T) = \sum_{\mu=4}^{4} P\left(\Sigma=\mu \,|\, e_m=0\right) + p\sum_{\mu=1}^{3} P\left(\Sigma=\mu \,|\, e_m=0\right) \quad /T=3 \longrightarrow$$

$$P_d = P\left(\Sigma=4 \,|\, e_m=0\right) + p\left[P\left(\Sigma=1 \,|\, e_m=0\right)+P\left(\Sigma=2 \,|\, e_m=0\right)+P\left(\Sigma=3 \,|\, e_m=0\right)\right]$$

$$\longrightarrow \quad P_d = (1-p)P\left(\Sigma=4 \,|\, e_m=0\right) + p\left[1-P\left(\Sigma=0 \,|\, e_m=0\right)\right]$$

$$\longrightarrow \quad P_d(3) = 2.200\times10^{-7} \qquad \text{(D)}$$

Finally, from (6.21): $P_d(4) = 10^{-4}$ (E)

∎

APPENDIX 6.2: CAPACITY OF THE BINARY SYMMETRIC CHANNEL

In this appendix, the term channel capacity will be introduced, together with any other related concepts. The *channel* between two devices (usually between a pair of complementary devices like encoder & decoder, modulator & demodulator, etc) is understood to mean the collection of hardware and physical media between the O/P of the 1st and the I/P of the 2nd. In communications, channels are usually 'noisy', i.e. they distort the message signal in a random fashion. This undesirable effect destroys some of the information contained in the message signal. It becomes clear, therefore, that a measure is required, of the amount of information about the message signal contained in the observed O/P of the channel. Shannon defined the concept of mutual information between events A & B:

Definition A6.2.1: The *mutual information* between events A & B, denoted by I(A;B), is the information provided about event A by the occurence of event B.

∎

A measure for I(A;B) should satisfy the following two intuitive properties (see, for example, Viterbi & Omura, [26]):

i) If A & B are independent events, then the occurence of B should provide no information about A.
ii) If the occurence of B indicates that A has definitely occured, then the occurence of B should provide us with all the information about A.

With the above specifications in mind, the following measure is proposed:

$$I(A;B) \triangleq \log P(A|B)/\log P(A) \qquad (A6.2.1)$$

Consider now a discrete memoryless channel (DMC - see Paragraph 1.1.4.) with input alphabet X, output alphabet Y and conditional probabilities $P(y|x)$, where the y_s are letters of the O/P alphabet and the x_s of the I/P alphabet. Let furthermore $q(x)$ denote the probability of occurence of the I/P letter x.

The main interest, with respect to a channel, is the average amount of information, the O/P of the channel provides, about the I/P.

Definition A6.2.2: The *average mutual information* between inputs and outputs of the DMC is defined to be:

$$I(X;Y) \triangleq E\left[I(x;y)\right] \qquad (A6.2.2)$$

∎

$I(X;Y)$ is defined in terms of the $P(y|x)_s$ and $q(x)_s$. It is possible to maximize I($X;Y$), over all I/P letter probability distributions, q(x):

<u>Definition A6.2.3</u>: The *channel capacity* of a DMC is defined to be the maximum average mutual information, where the maximization is over all possible input probability distributions:

$$C \triangleq \underset{q}{\mathrm{MAX}}\left\{I(X;Y)\right\} \qquad \text{(A6.2.3)}$$

∎

By symmetry, the capacity of the BSC, is achieved when its two inputs are equally probable [q(0)=q(1)=1/2]. Then it can be shown that (see for example Viterbi & Omura, [26]) if p is the channel error probability:

$$C_{BSC} = 1 + p\log_2 p + (1-p)\log_2(1-p) \text{ bits/symbol} \qquad \text{(A6.2.4)}$$

When using the BSC (or any channel for that matter), one has to take into account the maximum permissible code rate R. Specifically, in assessing the performance of a rate-R code, for various channel error probabilities p, one should not exceed the channel capacity C, or for a given p one should not use codes with rate R>C (see Theorem 1.3). Hence:

$$R < 1 + p\log_2 p + (1-p)\log_2(1-p) \qquad \text{(A6.2.5)}$$

The maximum code rate for various channel error probabilities, p, is given below:

<u>TABLE A6.2.1</u>

p	R_{max}	p	R_{max}
0.000001	0.99998	0.010000	0.91921
0.000010	0.99982	0.020000	0.85856
0.000100	0.99853	0.050000	0.71360
0.001000	0.98859	0.070000	0.63408
0.002000	0.97919	0.100000	0.53100
0.005000	0.95459	0.200000	0.27807
0.007000	0.93983	0.500000	0

TABLE A6.2.2

R	P_{max}	R	P_{max}
1/10	3.160×10^{-1}	7/8	1.713×10^{-2}
1/9	3.063×10^{-1}	8/9	1.479×10^{-2}
1/8	2.949×10^{-1}	9/10	1.299×10^{-2}
1/7	2.812×10^{-1}	10/11	1.155×10^{-2}
1/6	2.644×10^{-1}	11/12	1.038×10^{-2}
1/5	2.430×10^{-1}	12/13	9.420×10^{-3}
1/4	2.145×10^{-1}	13/14	8.610×10^{-3}
1/3	1.740×10^{-1}	14/15	7.920×10^{-3}
1/2	1.100×10^{-1}	15/16	7.327×10^{-3}
2/3	6.149×10^{-2}	16/17	6.812×10^{-3}
3/4	4.169×10^{-2}	29/30	3.468×10^{-3}
4/5	3.112×10^{-2}	49/50	1.910×10^{-3}
5/6	2.462×10^{-2}	99/100	8.602×10^{-4}
6/7	2.025×10^{-2}	999/1000	6.515×10^{-5}

APPENDIX 6.3: STUDY OF H(p,1)/H(p,c)

$$\text{Let } F(p,c) \mathrel{\hat{=}} H(p,1)/H(p,c) \qquad \text{(A6.3.1)}$$

From eqn (6.34b), with

$$P \mathrel{\hat{=}} 1-2p \qquad \text{(A6.3.2)}$$

$$H(p,c) = \ln\left[(1-P^c)/(1+P^c)\right] \qquad \text{(A6.3.3)}$$

At first, the two derivatives of H, dH/dp & dH/dc, will be calculated:

$$dH/dp = (dH/dP)(dP/dp) \longrightarrow dH/dp =$$

$$= \left\{\left[(1+P^c)/(1-P^c)\right]\left[(-cP^{c-1})(1+P^c)-(1-P^c)(cP^{c-1})\right]\Big/(1+P^c)^2\right\}(-2)$$

$$\longrightarrow dH/dp = \left\{\left[(1+P^c)/(1-P^c)\right](-cP^{c-1})(1+P^c+1-P^c)/(1+P^c)^2\right\}(-2)$$

$$\longrightarrow dH/dp = 4cP^{c-1}/(1-P^{2c}) \qquad \text{(A6.3.4)}$$

Also, since $P^c = e^{c\ln P} \longrightarrow dP^c/dc = e^{c\ln P}\ln P = P^c\ln P$:

$$dH/dc = \left\{\left[(1+P^c)/(1-P^c)\right]\left[-(1+P^c)-(1-P^c)\right]\Big/(1+P^c)^2\right\}P^c\ln P$$

$$\longrightarrow dH/dc = -P^c\ln P(1+P^c+1-P^c)/(1-P^{2c})$$

$$\longrightarrow dH/dc = -2P^c\ln P/(1-P^{2c}) \qquad \text{(A6.3.5)}$$

The following two theorems are concerned with the variation of F(p,c) with p & c:

<u>Theorem A6.3.1</u>: F(p,c) is a continuously increasing function of p, for $0<p<0.5$.

<u>Proof</u>: From (A6.3.1) & (A6.3.4):

$$dF/dp = d\left[H(p,1)/H(p,c)\right]/dp$$

$$= \left\{H(p,c)\left[dH(p,1)/dp\right] - H(p,1)\left[dH(p,c)/dp\right]\right\}/\left[H(p,c)\right]^2$$

$$\Longrightarrow \quad dF/dp = \ln\left[(1-P^c)/(1+P^c)\right]4/\left(1-P^2\right)/\left[H(p,c)\right]^2 -$$

$$- \ln\left[(1-P)/(1+P)\right]4cP^{c-1}/\left(1-P^{2c}\right)/\left[H(p,c)\right]^2$$

$$\Longrightarrow \quad \left(dF/dp\right)/\left\{4/\left[H(p,c)\right]^2\right\} =$$

$$\ln\left[(1-P^c)/(1+P^c)\right]/\left(1-P^2\right) - \ln\left[(1-P)/(1+P)\right]cP^{c-1}/\left(1-P^{2c}\right) \qquad (A)$$

Obviously, the sign of dF/dp is the sign of the RHS of eqn (A). An inequality will be constructed that will determine this sign:

Let $0<q<1 \Longrightarrow 0<q^k<1$ for $k=1,2,\ldots$

$$\Longrightarrow \quad 0 < \sum_{k=1}^{c-1} q^k < c-1 \quad /c>1 \quad \Longrightarrow \quad \sum_{k=0}^{c-1} q^k < c$$

$$\Longrightarrow \quad (1-q)\sum_{k=0}^{c-1} q^k < c(1-q) \quad \Longrightarrow \quad \sum_{k=0}^{c-1} q^k - \sum_{k=1}^{c} q^k < c(1-q) \quad \Longrightarrow$$

$$1-q^c < c(1-q) \quad \Longrightarrow \quad c(1-q)/(1-q^c) > 1 \quad /0<q<1 \;\&\; c>1 \qquad (B)$$

Also, for $0<q<1$ & $k=0,1,\ldots$: $\qquad q^{k(c-1)} \le 1 \qquad$ (C)

From (B) & (C), for $q=P^2$ $(0<P<1)$:

$$c(1-P^2)/(1-P^{2c}) > 1 \ge P^{2k(c-1)}$$

$$\Longrightarrow \quad cP^{c+2k}(1-P^2)/(1-P^{2c}) > P^{2kc-2k+c+2k} = (P^c)^{2k+1}$$

$$\Longrightarrow \quad c\left[(1-P^2)/(1-P^{2c})\right]P^{c-1}P^{2k+1} > (P^c)^{2k+1} \quad /k=0,1,\ldots \quad \Longrightarrow$$

$$c\left[(1-P^2)/(1-P^{2c})\right]P^{c-1}\left[P^{2k+1}/(2k+1)\right] > (P^c)^{2k+1}/(2k+1) \quad /k=0,1,\ldots$$

$$\Longrightarrow \quad 2\sum_{k=0}^{+\infty} c\left[(1-P^2)/(1-P^{2c})\right]P^{c-1}\left[P^{2k+1}/(2k+1)\right] >$$

$$> 2\sum_{k=0}^{+\infty}(P^c)^{2k+1}/(2k+1)$$

$$\Longrightarrow \quad c(1-P^2)/(1-P^{2c})P^{c-1}\sum_{k=0}^{+\infty}2P^{2k+1}/(2k+1) >$$

$$> \sum_{k=0}^{+\infty}2(P^c)^{2k+1}/(2k+1)$$

From Kreyszig [40], p. 580:

$$\sum_{k=0}^{+\infty}2z^{2k+1}/(2k+1) = \ln\left[(1+z)/(1-z)\right] \quad /|z|<1$$

Then, since $0<P<1$ & $0<P^c<1$ $/c>1$:

$$c\left[(1-P^2)/(1-P^{2c})\right]P^{c-1}\left\{-\ln\left[(1-P)/(1+P)\right]\right\} > -\ln\left[(1-P^c)/(1+P^c)\right]$$

$$\Longrightarrow \quad \left[1/(1-P^2)\right]\ln\left[(1-P^c)/(1+P^c)\right] >$$

$$> \left[cP^{c-1}/(1-P^{2c})\right]\ln\left[(1-P)/(1+P)\right]$$

for $0<P<1 \iff 0<1-2p<1 \iff 0<p<0.5$.

From the last result and eqn (A):

$dF/dp > 0$ for $0<p<0.5$.

QED

<u>Theorem A6.3.2</u>: $F(p,c)$ is a continuously increasing function of c, $c\geq 1$.

<u>Proof</u>: From eqn (A6.3.1),

$$dF(p,c)/dc = H(p,1)(-1)[H(p,c)]^2 dH(p,c)/dc =$$

$$= -H(p,1)[H(p,c)]^2[dH(p,c)/dc]$$

From eqn (A6.3.3), $H(p,1) < 0$

From eqn (A6.3.5), $dH(p,c)/dc > 0$

Then, $dF(p,c)/dc > 0$.

QED

Consider now the limit value of $F(p,c)$ as $p \longrightarrow 0$.

From eqns (A6.3.1), (A6.3.2) & (A6.3.3):

$$F(p,c) = \ln\left[p/(1-p)\right]/\ln K \quad \longrightarrow$$

$$\longrightarrow \quad \lim_{p\to 0}\left\{F(p,c)\right\} = \ln\left\{\lim_{p\to 0}\left[p/(1-p)\right]\right\}/\ln\left[\lim_{p\to 0}(K)\right] \qquad \text{(D)}$$

From eqns (D) & (6.28):

$$\lim_{p\to 0}\left\{F(p,c)\right\} = \ln(p)/\ln(pc) = 1/(1+\ln c/\ln p) \quad /pc \ll 1 \qquad \text{(E)}$$

Since, by Theorems A6.3.1 & A6.3.2, $F(p,c)$ is a continuously increasing function of p & c,

$$F(p,c) \geq F(p_{min},c_{min}) = F(p_{min},1) = 1$$

Hence, $F(p,c) \geq 1$; the following lemma has been proved:

<u>Lemma A6.3.1</u>: $F(p,c) \geq 1$ (A6.3.6a)

$$\lim_{p\to 0}\left\{F(p,c)\right\} = 1/(1+\ln c/\ln p) \quad /pc \ll 1 \qquad \text{(A6.3.6b)}$$

∎

Consider now the limit values of $F(p,c)$, as $c \longrightarrow +\infty$ and $p \longrightarrow 0.5$.

For p constant and $0<p<0.5$, $P=1-2p$ is also constant and $0<P<1$. Then $\ln P<0$ and hence,

$$P^c = e^{c\ln P} \longrightarrow 0 \quad \text{as} \quad c \longrightarrow +\infty \qquad \text{(F)}$$

Then, $1-P^c \longrightarrow 1_-$ as $c \longrightarrow +\infty$

and, $1+P^c \longrightarrow 1_+$ as $c \longrightarrow +\infty$

Hence, $(1-P^c)/(1+P^c) \longrightarrow 1_-/1_+$ as $c \longrightarrow +\infty$

and $H(p,c) = \ln\left[(1-P^c)/(1+P^c)\right] \longrightarrow 0_-$ as $c \longrightarrow +\infty$

Finally: $F(p,c) = H(p,1)/H(p,c) \longrightarrow +\infty$ as $c \longrightarrow +\infty$ (G)

Let c constant $/c>1$. As $p \longrightarrow 0.5$, $P \longrightarrow 0$. Then:

$H(p,c) = \ln\left[(1-P^c)/(1+P^c)\right] \longrightarrow 0$ as $P \longrightarrow 0$, so

$F(p,c) = H(p,1)/H(p,c) \longrightarrow 0/0$ as $P \longrightarrow 0$.

Using the derivatives of H(p,1) & H(p,c) [from eqn (A6.3.4)]:

$$\lim_{p\to 0}\left\{F(p,c)\right\} = \lim_{p\to 0}\left\{\left[dH(p,1)/dp\right]/\left[dH(p,c)/dp\right]\right\}$$

$$= \lim_{p\to 0}\left\{\left[4/(1-P^2)\right]/\left[4cP^{c-1}/(1-P^{2c})\right]\right\}$$

$$= \lim_{p\to 0}\left\{(1-P^{2c})/(1-P^2)\right\}\lim_{p\to 0}\left\{1/cP^{c-1}\right\} = +\infty$$

Hence, the following theorem has been proved:

<u>Theorem A6.3.3</u>:

$$F(p,c) \longrightarrow +\infty \quad \text{as} \quad c \longrightarrow +\infty \qquad \text{(A6.3.7a)}$$

$$F(p,c) \longrightarrow +\infty \quad \text{as} \quad p \longrightarrow 0.5 \qquad \text{(A6.3.7b)}$$

∎

Consider now the range of values of F. Note that $T_o = \lfloor J/2+F/2 \rfloor$. Hence, as p increases from very small values, F also increases and T_o increases in steps of 1. Since F/2>0.5 (see Lemma A6.3.1), the values of interest of F are those for which F/2=k, where k=1,1.5,2,2.5,... Let F/2=k. Then from eqn (A6.3.1):

$$H(p,1)/H(p,c) = 2k \qquad \text{(H)}$$

Eqn (H) is very difficult (if not impossible) to solve analytically for p. So, it will be solved for c. From (H):

$$H(p,c) = H(p,1)/2k \quad \longrightarrow \quad \ln\left[(1-P^c)/(1+P^c)\right] = H(p,1)/2k$$

$$\longrightarrow \quad (1-P^c)/(1+P^c) = \exp\left[H(p,1)/2k\right] \triangleq A \qquad \text{(I)}$$

$$\longrightarrow \quad 1-P^c = A+AP^c \quad \longrightarrow \quad 1-A = P^c+AP^c \quad \longrightarrow$$

$$P^c = (1-A)/(1+A) \quad \longrightarrow \quad c = \ln\left[(1-A)/(1+A)\right]/\ln P \qquad \text{(J)}$$

From eqn (I): $A \triangleq \exp\left[H(p,1)/2k\right] = \exp\left\{\ln\left[p/(1-p)\right]/2k\right\}$

$\longrightarrow \quad A = [p/(1-p)]^{(1/2k)}$. Hence:

<u>Theorem A6.3.4</u>: If $A \triangleq [p/(1-p)]^{(1/2k)}$, the value of c which makes F/2=k /k=1,1.5,2,2.5,... is given by:

$$c = \ln\left[(1-A)/(1+A)\right]/\ln(1-2p) \qquad \text{(A6.3.8)}$$

∎

Consider now an approximate solution of F/2=1. From eqn (A6.3.1):

$$F = 2 \quad \Longrightarrow \quad \ln\left[(1-P)/(1+P)\right] = 2\ln\left[(1-P^c)/(1+P^c)\right]$$

$$\Longrightarrow \quad (1-P)/(1+P) = (1-P^c)^2/(1+P^c)^2 = (1+P^{2c}-2P^c)/(1+P^{2c}+2P^c)$$

$$\Longrightarrow \quad 1+P^{2c}+2P^c-P-P^{2c+1}-2P^{c+1} = 1+P^{2c}-2P^c+P+P^{2c+1}-2P^{c+1}$$

$$\Longrightarrow \quad P^{2c}-2P^{c-1}+1 = 0 \qquad (K)$$

Consider now an approximation for P^n:

$$P^n = (1-2p)^n = \sum_{i=0}^{n}(-2p)^i\binom{n}{i} \qquad (L)$$

Because p is very small ($p \ll 1$), P^n will be approximated by the first three terms of the above summation. From (L):

$$(1-2p)^n \approx 1 - 2p\binom{n}{1} + 4p^2\binom{n}{2} = 1 - 2pn + 4p^2n(n-1)/2$$

$$\Longrightarrow \quad (1-2p)^n \approx 1 - 2pn + 4p^2n(n-1)/2 \qquad (A6.3.9)$$

From (A6.3.9) & (K):

$$1 - 4pc + 4p^2(2c^2-c) - 2 + 4p(c-1) - 4p^2(c^2-3c+2) + 1 \approx 0$$

$$\Longrightarrow \quad -4p(c-c+1) + 4p^2(2c^2-c-c^2+3c-2) \approx 0$$

$$\Longrightarrow \quad p(c^2+2c-2) \approx 1 \quad \Longrightarrow \quad p \approx 1/(c^2+2c-2) \quad \Longrightarrow$$

$$p \approx 1/(c^2+2c) \approx 1/c^2$$

Hence:

<u>Lemma A6.3.2</u>: The value of p which makes F=2, is $p \approx 1/c^2$.

∎

APPENDIX 6.4: <u>PROOF OF RELATION (6.38c)</u>

From (6.38b):

$$P_d(T_o) = Q^J\left[\sum_{\mu=T_o+1}^{J}K^\mu\binom{J}{\mu} + p\sum_{\mu=J-T_o}^{T_o}K^\mu\binom{J}{\mu}\right] \qquad (6.38b)$$

$$\mathbb{P}_d(T_o) = Q^J\sum_{\mu=To+1}^{J}K^\mu\binom{J}{\mu} + pQ^J\Big[\sum_{\mu=0}^{J}K^\mu\binom{J}{\mu} - \sum_{\mu=0}^{J-To-1}K^\mu\binom{J}{\mu} - \sum_{\mu=To+1}^{J}K^\mu\binom{J}{\mu}\Big] \longrightarrow$$

$$\mathbb{P}_d(T_o) = p\sum_{\mu=0}^{J}Q^J K^\mu\binom{J}{\mu} - Q^J\Big[p\sum_{\mu=0}^{J-T_o-1}K^\mu\binom{J}{\mu} - (1-p)\sum_{\mu=T_o+1}^{J}K^\mu\binom{J}{\mu}\Big] \qquad \text{(A)}$$

Let $\tau = J-\mu$, in the last summation in the RHS of (A). Note, also, that $C(J,J-\mu) = C(J,\mu)$:

$$\mathbb{P}_d(T_o) = p\sum_{\mu=0}^{J}Q^{J-\mu}P^\mu\binom{J}{\mu} - Q^J\Big[p\sum_{\mu=0}^{J-To-1}K^\mu\binom{J}{\mu} - (1-p)\sum_{\tau=0}^{J-To-1}K^{J-\tau}\binom{J}{\tau}\Big] \longrightarrow$$

$$\mathbb{P}_d(T_o) = p(P+Q)^J - Q^J\sum_{\mu=0}^{J-To-1}\binom{J}{\mu}\big[pK^\mu - (1-p)K^{J-\mu}\big] \longrightarrow$$

$$\mathbb{P}_d(T_o) = p - Q^J\sum_{\mu=0}^{J-T_o-1}\binom{J}{\mu}\big[pK^\mu - (1-p)K^{J-\mu}\big] \qquad \text{(B)}$$

Consider now the sign of the quantity in brackets, in the summation in the RHS of (B):

From Theorem 6.5 (F is defined in Theorem 6.6):

$$T_o = \lfloor(J+F)/2\rfloor \le (J+F)/2 < T_o+1$$

$$\longrightarrow \quad T_o \le (J+F)/2 < T_o+1 \qquad \text{(A6.4.1)}$$

$$\longrightarrow \quad J-T_o-1 < (J-F)/2 \le J-T_o \qquad \text{(A6.4.2)}$$

For $0 \le \mu \le J-T_o-1$:

$$0 \le \mu \le J-T_o-1 < (J-F)/2 \longrightarrow 2\mu < J-F \longrightarrow F < J-2\mu$$

$$\longrightarrow \ln[p/(1-p)]/\ln K < J-2\mu \longrightarrow \ln[p/(1-p)] > \ln K^{J-2\mu}$$

$$\longrightarrow p/(1-p) > K^{J-2\mu} = K^{J-\mu}/K^\mu \longrightarrow pK^\mu > (1-p)K^{J-\mu}$$

$$\longrightarrow \text{For } 0 \le \mu \le J-T_o-1: \quad pK^\mu-(1-p)K^{J-\mu} > 0 \qquad \text{(C)}$$

Hence, from (B) & (C): $\mathbb{P}_d(T_o) < p$ for $T_o < J$

QED

APPENDIX 6.5: GENERALIZED MEANS

A6.5.1. Proof of Theorem 6.9

From Definition 6.1, for $\mu=1$ & $\mu=J$:

$$A_1 \triangleq \left\{\left[\sum_{x(1)=1}^{J-1+1} K_{x(1)} \sum_{x(2)=x(1)+1}^{J-1+2} K_{x(2)} \cdots \sum_{x(1)=x(1-1)+1}^{J} K_{x(1)}\right] \Big/ \binom{J}{1}\right\}^{1/1} \longrightarrow$$

$$\longrightarrow \quad A_1 = (1/J) \sum_{x(1)=1}^{J} K_{x(1)} = \text{arithmetic mean}$$

$$A_J \triangleq \left\{\left[\sum_{x(1)=1}^{J-J+1} K_{x(1)} \sum_{x(2)=x(1)+1}^{J-J+2} K_{x(2)} \cdots \sum_{x(J)=x(J-1)+1}^{J} K_{x(J)}\right] \Big/ \binom{J}{J}\right\}^{1/J} \longrightarrow$$

$$\longrightarrow \quad A_J = (K_1 K_2 \cdots K_J)^{1/J} = \text{geometric mean}$$

It is known that the arithmetic mean of J positive numbers is always greater than their geometric mean, if these J numbers are not all the same (see Barnard & Child [41]).

Then, since $(A_\mu)^\mu$ is the arithmetic mean of $C(J,\mu)$ numbers [the $C(J,\mu)$ products $K_{x(1)}K_{x(2)}\cdots K_{x(\mu)}$], $(A_\mu)^\mu$ is greater than their geometric mean. The latter is the $C(J,\mu)$th root of the product of $C(J,\mu)$ distinct products of μ Ks.

Given any specific K_i ($1 \le i \le J$), there are $C(J-1,\mu-1)$ distinct ways to form a product of μ Ks, hence as many distinct products of μ Ks that include K_i. Hence each specific K_i appears $C(J-1,\mu-1)$ times in these products. Then:

$$(A_\mu)^\mu > \left[(K_1 K_2 \cdots K_J)^{C(J-1,\mu-1)}\right]^{1/C(J,\mu)} = (K_1 K_2 \cdots K_J)^{\mu/J}$$

because:

$$\binom{J-1}{\mu-1} \Big/ \binom{J}{\mu} = \Big[(J-1)!\,\mu!\,(J-\mu)!\Big] \Big/ \Big[J!\,(\mu-1)!\,(J-1-\mu+1)!\Big] = \mu/J$$

Then: $\quad A_\mu > (K_1 K_2 \cdots K_J)^{1/J} = A_J$

QED

A6.5.2. Proof of Theorem 6.10

The following, forms the basis of Theorem 6.10.

<u>Theorem A6.5.1</u>: Consider J positive real numbers K_1, K_2

$,\ldots,K_J$, and all $C(J,\mu-1)$ distinct products of $\mu-1$ K_is, as well as all $C(J,\mu)$ distinct products of μ K_is. Let collection C1 be made of $J-\mu+1$ replicas of the products of $\mu-1$ K_is and collection C2 be made of μ replicas of the products of μ K_is. Then, the two collections have the same number of elements, and for each element, $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}]$, from C1 there is an element in C2, of the form $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}K_y]$, where $1\le x(1)<x(2)<\cdots<x(\mu-1)\le J$, while $y \ne x(i)$ /$i=1,2,\ldots,\mu-1$ & $1\le y\le J$.

<u>Proof</u>: C1 is made of $J-\mu+1$ replicas of $C(J,\mu-1)$ distinct elements, while C2 is made of μ replicas of $C(J,\mu)$ distinct elements.

$$(J-\mu+1)\binom{J}{\mu-1} = (J-\mu+1)J!/[(\mu-1)!(J-\mu+1)!] =$$

$$= J!/[(\mu-1)!(J-\mu)!] = \mu J!/[\mu!(J-\mu)!] = \mu\binom{J}{\mu} \qquad \text{(A)}$$

From eqn (A), collections C1 & C2 contain the same number of elements.

A method will be proposed to generate C2 from C1.

C1 contains $J-\mu+1$ $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}]$s. Multiply each of the identical $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}]$s by a K_z other than $K_{x(1)},K_{x(2)},\ldots,K_{x(\mu-1)}$ [i.e. $z \ne x(i)$ /$i=1,2,\ldots,\mu-1$]. There are exactly $J-\mu+1$ such K_zs, hence each $[K_{x(1)},K_{x(2)},\ldots,K_{x(\mu-1)}]$ generates $J-\mu+1$ distinct elements of C2. Hence, since C1 has $C(J,\mu-1)$ distinct elements, $(J-\mu+1)C(J,\mu-1) = \mu C(J,\mu)$ elements of C2 are generated. For the generated collection to be C2, though, it must contain exactly μ copies of each distinct product of μ Ks.

Consider elements $[K_{y(2)}\cdots K_{y(\mu-1)}K_{y(\mu)}]$, $[K_{y(1)}\cdots K_{y(\mu-1)}K_{y(\mu)}],\ldots,$ $[K_{y(1)}K_{y(2)}\cdots K_{y(\mu)}]$, $[K_{y(1)}K_{y(2)}\cdots K_{y(\mu-1)}]$ of C1. Multiply the 1st with $K_{y(1)}$, the 2nd with $K_{y(2)},\ldots,$the $(\mu-1)$th with $K_{y(\mu-1)}$ and the μth with $K_{y(\mu)}$. Hence, the generated collection contains at least μ copies of each of its elements.

<u>Assume</u> that there is at least one product of μ K_is, say $[K_{z(1)}K_{z(2)}\cdots K_{z(\mu)}]$, that does not belong to the generated collection. Then, all the μ products of $\mu-1$ $K_{z(i)}$s,

$[K_{z(2)}\cdots K_{z(\mu-1)}K_{z(\mu)}]$, $[K_{z(1)}\cdots K_{z(\mu-1)}K_{z(\mu)}]$, ..., $[K_{z(1)}K_{z(2)}\cdots K_{z(\mu)}]$, $[K_{z(1)}K_{z(2)}\cdots K_{z(\mu-1)}]$

cannot belong to C1, because multiplication of any of them by the appropriate $K_{z(i)}$ $[K_{z(1)}, K_{z(2)}, \ldots, K_{z(\mu-1)}, K_{z(\mu)}$, respectively], would have generated $K_{z(1)}K_{z(2)}\cdots K_{z(\mu)}$. But this contradicts the fact that C1 contains all possible products of μ-1 K_is. Hence, all the $C(J,\mu)$ distinct products of μ K_is are contained in the generated collection and, according to a previous conclusion, at least μ copies of each.

Then, the generated collection contains at least $\mu C(J,\mu)$ elements but since $(J-\mu+1)C(J,\mu-1) = \mu C(J,\mu)$ elements were generated, it contains exactly μ copies of each of the $C(J,\mu)$ distinct products of μ K_is; hence the generated collection is C2.

QED

According to the generation rule of the proof of Theorem A6.5.1, each $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}]$ of C1 is multiplied with the $J-\mu+1$ K_is which belong to

$$\left\{K_1, K_2, \ldots, K_J\right\} - \left\{K_{x(1)}, K_{x(2)}, \ldots, K_{x(\mu-1)}\right\}$$

Hence the sum of the elements of C2 that are generated from the $J-\mu+1$ $[K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)}]$s may be expressed by:

$$K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)} \sum_{\substack{x(\mu) \\ x(\mu)\neq x(i) \\ 0<i<\mu}} K_{x(\mu)}$$

Hence, the sum of the elements of C2 may be written as

$$\sum_{x(1)=1}^{J-\mu+2} \; \sum_{x(2)=x(1)+1}^{J-\mu+3} \cdots \sum_{x(\mu-1)=x(\mu-2)+1}^{J} K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)} \sum_{\substack{x(\mu) \\ x(\mu)\neq x(i) \\ 0<i<\mu}} K_{x(\mu)} \qquad \text{(B)}$$

The sum of the elements of C1 is nothing more than the sum of the elements of all the distinct products of μ-1 K_is [$C(J,\mu-1)$ such products] multiplied by $J-\mu+1$. Since, in the above multiple summation, the last sum is over $J-\mu+1$ factors, the sum of the elements of C1 may be expressed by:

$$\sum_{x(1)=1}^{J-\mu+2} \sum_{x(2)=x(1)+1}^{J-\mu+3} \cdots \sum_{x(\mu-1)=x(\mu-2)+1}^{J} K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)} \sum_{\substack{x(\mu) \\ x(\mu)\neq x(i) \\ 0<i<\mu}} 1 \qquad \text{(C)}$$

Note though that the sum of the elements of C1, if divided by $J-\mu+1$, gives the sum of all distinct products of $\mu-1$ K_is; if it is further divided by $C(J,\mu-1)$ it gives $(A_{\mu-1})^{\mu-1}$. Similarly, the sum of the elements of C2 divided by $\mu C(J,\mu)$ gives $(A_\mu)^\mu$. Hence, the difference of summation (C) minus summation (B) equals $(J-\mu+1)C(J,\mu-1)(A_{\mu-1})^{\mu-1} - \mu C(J,\mu)(A_\mu)^\mu$ or, using eqn (A), $\mu C(J,\mu)[(A_{\mu-1})^{\mu-1}-(A_\mu)^\mu]$:

$$\mu\binom{J}{\mu}\left[(A_{\mu-1})^{\mu-1} - (A_\mu)^\mu\right] =$$

$$= \sum_{x(1)=1}^{J-\mu+2} \sum_{x(2)=x(1)+1}^{J-\mu+3} \cdots \sum_{x(\mu-1)=x(\mu-2)+1}^{J} K_{x(1)}K_{x(2)}\cdots K_{x(\mu-1)} \sum_{\substack{x(\mu) \\ x(\mu)\neq x(i) \\ 0<i<\mu}} [1-K_{x(\mu)}] \qquad \text{(D)}$$

Note, from eqn (D), that if all K_is are less than, or equal to, 1 with at least one $K_i<1$ then the RHS is positive. Similarly, if all K_is are ≥ 1 with at least one $K_i>1$ then the RHS of eqn (D) is negative.

QED

APPENDIX 6.6: OPTIMUM THRESHOLD FOR FEEDBACK DECODING

From eqns (6.3) & (6.4) and Lemma 6.1:

$$\delta\mathbf{P}_d(T) \triangleq \mathbf{P}_d(T) - \mathbf{P}_d(T-1) = pP(\Sigma=J-T) - (1-p)P(\Sigma=T)$$

Using eqn (6.44), in the above eqn:

$$\delta\mathbf{P}_d(T) = pQ(J)(A_{J-T})^{J-T}\binom{J}{J-T} - (1-p)Q(J)(A_T)^T\binom{J}{T} \longrightarrow$$

$$\delta\mathbf{P}_d(T) = Q(J)\binom{J}{T}\left[p(A_{J-T})^{J-T} - (1-p)(A_T)^T\right] \quad /0<T\leq J \qquad \text{(A)}$$

Note from eqn (A) that the sign of $\delta\mathbf{P}_d(T)$ is the sign of $p(A_{J-T})^{J-T} - (1-p)(A_T)^T$. The sign of a difference, say A-B, is positive if $A>B \iff A/B>1 \iff \ln(A/B)>0$, negative if

$\ln(A/B)<0$ and zero if $\ln(A/B)=0$. Hence, the sign of $\delta\mathbb{P}_d(T)$ is the sign of

$$E(T) \triangleq \ln\Big\{[p(A_{J-T})^{J-T}]/[(1-p)(A_T)^T]\Big\} \quad /0<T\leq J \qquad (A6.6.1)$$

$$\Longrightarrow \quad E(T) = \ln[p/(1-p)] + (J-T)\ln A_{J-T} - T\ln A_T \qquad (A6.6.2)$$

The following theorem has then been proved:

<u>Theorem A6.6.1</u>: Let J syndrome bits, with sizes c_i $/i=1,2,\ldots,J$ checking on error bit $e_h^{(a)}$ and K_i be defined by eqn (6.16). If p denotes the BSC's error probability and $\mathbb{P}_d(T)$ the probability that $e_h^{(a)}$ will be erroneously estimated, using a threshold T and FD, then the sign of $\mathbb{P}_d(T) - \mathbb{P}_d(T-1)$ is the sign of

$$E(T) = \ln[p/(1-p)] + (J-T)\ln A_{J-T} - T\ln A_T \qquad (A6.6.2)$$

where A_μ is the μth generalized mean of the J K_is $/\mu=1,2,\ldots,J$, $A_0=1$ and $0<T\leq J$.

∎

It is necessary to examine the behaviour of E(T). It will be shown that E(T) is a continuously increasing function of T and that $E(T)<0$ for $T<J/2$.

Consider the difference $E(T) - E(T-1)$. From eqn (A6.6.2):

$$E(T) - E(T-1) = \ln(A_{J-T})^{J-T} - \ln(A_T)^T -$$

$$- \ln(A_{J-T+1})^{J-T+1} + \ln(A_{T-1})^{T-1} \Longrightarrow$$

$$E(T) - E(T-1) = \ln\Big[(A_{J-T})^{J-T}/(A_{J-T+1})^{J-T+1}\Big] + \ln\Big[(A_{T-1})^{T-1}/(A_T)^T\Big]$$

From Theorem 6.10, and because $K_i = P_i/(1-P_i) < 1$:

$$(A_{J-T})^{J-T} > (A_{J-T+1})^{J-T+1} \quad \& \quad (A_{T-1})^{T-1} > (A_T)^T$$

Hence the arguments of both logarithms are >1. Then,

$$E(T) > E(T-1) \qquad (A6.6.3)$$

Consider now the sign of E(T). Since $p<1-p$, then $\ln[p/(1-p)] < 0$. E(T) will be negative if $\ln[(A_{J-T})^{J-T}/(A_T)^T] < 0$ or, the same, if $(A_{J-T})^{J-T} < (A_T)^T$. According to Theorem 6.10, this happens if $J-T > T \Longleftrightarrow T < J/2$. It follows then that if $E(T) = 0$ has a solution this will occur for $T \geq$

$J/2$ (this does not imply that $E(T) \geq 0$ for $T \geq J/2$).

Theorem A6.6.2: Let $E(T)$ be defined by Theorem A6.6.1. Then, $E(T)$ is a continuously increasing function of T ($0<T\leq J$). Furthermore, $E(T) < 0$, for $T<J/2$.

∎

Note, from Theorem A6.6.2, that $E(T)$ is definitely negative for $T < J/2$. This means that if $E(T)$ changes sign, within the range $[1,J]$, this will occur in the range $[J/2,J]$. The sign of $E(T)$ is also the sign of $\delta\mathbb{P}_d(T)$ (see Theorem A6.6.1). According to Theorem 6.1, if $E(T) < 0$ for $T < X$, $E(T) > 0$ for $T > X$ and, in case $E(T) = 0$ has a solution $E(X) = 0$, then the optimum threshold is $T_o = \lfloor X \rfloor$. Since $E(T)$ will not change sign in the range $(0,J/2)$, then the optimum threshold will be at least $J/2$, and since it has to be an integer, $T_o \geq \lceil J/2 \rceil$.

This proves the first part of Theorem 6.11.

Since the sign of $\delta\mathbb{P}_d(T)$ is the sign of $E(T)$:

$$\delta\mathbb{P}_d(T) < 0 \iff E(T) < 0$$

$$\iff \ln[p/(1-p)] + (J-T)\ln A_{J-T} - T\ln A_T < 0$$

$$\iff \ln[p/(1-p)] + J\ln A_{J-T} < T\ln A_T + T\ln A_{J-T} = T\ln(A_{J-T}A_T)$$

$$\iff T < \left\{\ln[p/(1-p)] + J\ln A_{J-T}\right\}/\ln(A_{J-T}A_T)$$

Hence,

$$\delta\mathbb{P}_d(T) < 0 \iff T < \left\{\ln[p/(1-p)] + J\ln A_{J-T}\right\}/\ln(A_{J-T}A_T)$$

Similarly,

$$\delta\mathbb{P}_d(T) > 0 \iff T > \left\{\ln[p/(1-p)] + J\ln A_{J-T}\right\}/\ln(A_{J-T}A_T)$$

If $\delta\mathbb{P}_d(T) = 0$ has a solution,

$$\delta\mathbb{P}_d(T) = 0 \iff T = \left\{\ln[p/(1-p)] + J\ln A_{J-T}\right\}/\ln(A_{J-T}A_T)$$

According to Theorem 6.1,

$$\lfloor\{\ln[p/(1-p)] + J\ln A_{J-T}\}/\ln(A_{J-T}A_T)\rfloor$$

is the optimum threshold for the above case. Note though that the expression above is a function of T_o itself, hence it does not give T_o, but it has to be solved for T_o. Also, since $T_o \leq J$, T_o should not be allowed to exceed J.

This completes the proof of Theorem 6.11.

APPENDIX 7.1 INTRODUCTION TO ARITHMETICAL FUNCTIONS

This appendix will introduce the reader to the basic definitions and theorems on the so-called arithmetical functions (like the Euler function, the Mobius function, the greatest common divisor, etc). The material is based on the excellent textbook by Tom Apostol, "Introduction to Analytic Number Theory" [44]. It is the opinion of the author that number theory becomes increasingly important for various branches of electronic engineering, and as such it should be incorporated into the syllabuses of relevant under- & post-graduate courses.

Unless otherwise stated, small latin & greek letters denote integers.

Definition A7.1.1: A real- or complex-valued function defined on the positive integers is called an *arithmetical function* or a *number-theoretic function*. [44]

∎

Definition A7.1.2: It is said that d *divides* n, and this is denoted by $d|n$, if there exists an integer c such that $n = cd$. It is also said that n is a *multiple* of d. $d\nmid n$ denotes that d does not divide n. [44]

∎

Definition A7.1.3: The *greatest common divisor* (gcd) of two integers α & b is a nonnegative common divisor of α & b, denoted by (α,b), such that any other common divisor of α & b also divides (α,b). It can be proved that for any α & b, (α,b) is unique. If $(\alpha,b) = 1$, α & b are said to be *relatively prime*. [44]

∎

Theorem A7.1.1: The gcd has the following properties:

Commutative law: $$(\alpha,b) = (b,\alpha) \qquad \text{(A7.1.1a)}$$

Associative law: $$(\alpha,(b,c)) = ((\alpha,b),c) \qquad \text{(A7.1.1b)}$$

Distributive law: $$(ac,bc) = |c|(a,b) \qquad \text{(A7.1.1c)}$$

$$(a,1) = (1,a) = 1 \qquad \text{(A7.1.1d)}$$

$$(a,0) = (0,a) = |a| \qquad \text{(A7.1.1e)}$$

Proof: See Apostol [44], p. 16.

∎

Definition A7.1.4: If $n \geq 1$, the *Euler totient* $\Phi(n)$ is defined to be the number of positive integers not exceeding n which are relatively prime to n. [44]

∎

Theorem A7.1.2: *Fundamental theorem of arithmetic*: Every integer $n>1$ can be represented by a product of prime factors in only one way, apart from the order of the factors.

Proof: See Apostol [44], p. 17.

∎

Theorem A7.1.3: If $n = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r}$, where $a_i \geq 1$ for i=1, 2,...,r, then the set of positive divisors of n is the set of numbers of the form $p_1^{c_1} p_2^{c_2} \cdots p_r^{c_r}$, where $0 \leq c_i \leq a_i$ for i=1,2,...,r.

Proof: See Apostol [44], p. 18.

∎

Theorem A7.1.4: If two positive integers a and b have the factorization

$$a = \prod_{i=1}^{+\infty} p_i^{a_i}$$

$$b = \prod_{i=1}^{+\infty} p_i^{b_i}$$

where $a_i \geq 0$ & $b_i \geq 0$, then their gcd has the factorization

$$(a,b) = \prod_{i=1}^{+\infty} p_i^{c_i} \quad / c_i = \text{MIN}\{a_i, b_i\} \quad \text{for all } i \qquad \text{(A7.1.2)}$$

Proof: See Apostol [44], p. 18.

∎

Note that in the last theorem the products are over all prime numbers, but of course the products themselves are finite. So, $p_1=2$, $p_2=3$, $p_3=5,\ldots$, $p_{15}=47,\ldots$ etc.

Theorem A7.1.5: For any two positive integers a & b:

$$(a,b) = d \iff (a/d,b/d) = 1 \qquad (A7.1.3)$$

Proof: From (A7.1.1c), $(a/d,b/d) = 1 \implies d(a/d,b/d) = d \implies (a,b) = d$ (because $d\geq0$, by Definition A7.1.3). Let now $(a,b) = d$. If at least one of a & b is 1, then $(a/d,b/d) = (a,b) = 1$. The theorem will be proved for the case where a & b are >1. Let a & b have the following factorization:

$$a = \prod_{i=1}^{+\infty} p_i^{a_i} \qquad (A)$$

$$b = \prod_{i=1}^{+\infty} p_i^{b_i} \qquad (B)$$

where $a_i\geq0$ & $b_i\geq0$. From Theorem A7.1.3, d will have the factorization:

$$d = \prod_{i=1}^{+\infty} p_i^{d_i} \quad /0\leq d_i\leq \text{MIN}\{a_i,b_i\} \text{ for all } i \qquad (C)$$

From (A), (B) & (C), one obtains the following factorizations:

$$a/d = \prod_{i=1}^{+\infty} p_i^{a_i-d_i} \quad a_i-d_i\geq0 \qquad (D)$$

$$b/d = \prod_{i=1}^{+\infty} p_i^{b_i-d_i} \quad b_i-d_i\geq0 \qquad (E)$$

From (D) & (E) and Theorem A7.1.4:

$$(a/d,b/d) = \prod_{i=1}^{+\infty} p_i^{c_i} \quad /c_i = \text{MIN}\{a_i-d_i,b_i-d_i\} \text{ for all } i \qquad (F)$$

Then, $(a/d,b/d)=1 \implies c_i=0$ for all i, from (F)

$\Longrightarrow$ $MIN\{a_i,b_i\}=d_i$ for all i, from (F)

$\Longrightarrow$ $(a,b)=d$, from Theorem A7.1.4.

QED

Theorem A7.1.6: If $n\geq 1$, then $(n,n-1) = 1$ (A7.1.4)

Proof: Let $d \hat{=} (n,n-1)$. Then, there exist integers a & b: $n=ad$ & $n-1=bd \Longrightarrow ad-1=bd \Longrightarrow (a-b)d=1 \Longrightarrow d|1. \Longrightarrow d\leq 1$. Since $ad>bd \Longrightarrow d\neq 0$, hence $d=1$.

QED

Theorem A7.1.7: If m is a positive integer, then as c 'runs' through the range [1,m], m/(m,c) 'runs' through all the positive divisors of m:

$d|m \Longleftrightarrow$ there exists c $/1\leq c\leq m$: $d=m/(m,c)$ (A7.1.5)

Proof: Let c $/1\leq c\leq m$. Then, if $d = m/(m,c) \Longrightarrow d(m,c) = m \Longrightarrow d|m$.

Let $d|m$. Then, $m=kd$, where $1\leq k\leq m$. Let $c=m-k=kd-k=k(d-1)$. From Theorem A7.1.6, $(d,d-1)=1 \Longrightarrow (m/k,c/k)=1 \Longrightarrow (m,c)=k$ (from Theorem A7.1.5) $\Longrightarrow (m,c)=m/d \Longrightarrow d=m/(m,c)$.

QED

Theorem A7.1.8: Let b be a positive integer and p its smallest prime factor. Then, if $1\leq c<p$, $(b,c)=1$.

Proof: Let b, p & c, as above and $d \hat{=} (b,c)$. Since $d|c \Longrightarrow d\leq c$ and because $c<p \Longrightarrow d<p$. Assume that $d>1$. Let q be a prime factor of d. Since $(b,c)=d|b$, q is also a prime factor of b. But $q\leq d<p$, hence q is a prime factor of b, smaller than p. This contradicts the hypothesis, hence $d\leq 1$. Since, by Definition A7.1.3, d is nonnegative then $d=0$ or $d=1$. By Definitions A7.1.2 & A7.1.3, there exist integers x & y such that $b=xd$ & $c=yd$. Since b & c are not zero, by hypothesis, x, y & d must also be non-zero, hence $d=1$.

QED

Theorem A7.1.9: If a prime p does not divide a, then $(p,a)=1$.

Proof: See Apostol [44], p. 17.

∎

Theorem A7.1.10: If $a|bc$ and if $(a,b)=1$, then $a|c$.

Proof: See Apostol [44], p. 16.

∎

Theorem A7.1.11: For any integers a & b and any positive integers k & n:

$$(a,b) = 1 \implies (a^k,b^n) = 1 \qquad (A7.1.6)$$

Proof: Let a, b, k & n as defined above and $(a,b)=1$. Let $f \triangleq (a^k,b^n)$. Assume that $f>1$. Then, there must exist a prime p which divides both a^k & b^n.

Let $q_1,q_2,\ldots,q_r$ be the prime factors of an integer c. Then c^m has the same prime factors (except that they are all raised to power m). Hence, a^k has the same prime factors with a and b^n has the same prime factors with b. So p is a prime factor of both a and b and $(a,b) \geq p$, which contradicts $(a,b)=1$. Hence, $(a^k,b^n) = 1$.

QED

Theorem A7.1.12: For any integers a, b & c:

$$(a+cb,b) = (a,b) \qquad (A7.1.7)$$

Proof: Let $(a,b) \triangleq h$ and $(a+cb,b) \triangleq f$. It will be shown that $f|h$ & $h|f$.

Since $(a+cb,b)=f$, $f|(a+cb)$ & $f|b$, hence there exist integers k & m, such that $a+cb=kf$ & $b=mf$. It follows that $a=kf-cmf \implies a=(k-cm)f \implies f|a$. Then $f|(a,b) \implies f|h$.

Since $(a,b)=h$, $h|a$ & $h|b$, there exist integers n & s such that $a=nh$ & $b=sh$. It follows that $a+cb = nh+csh = (n+cs)h \implies h|(a+cb)$. Then, $h|(a+cb,b)=f$.

QED

Theorem A7.1.13: For any a, b & c:

$$(a,b) = (a,c) = 1 \implies (a,bc) = 1 \qquad (A7.1.8)$$

Proof: Let the prime decomposition of a, b & c:

$$a = p_1^{a_1} p_2^{a_2} p_3^{a_3} \cdots \quad / a_i \ge 0 \text{ for } i=1,2,3,\ldots$$

$$b = p_1^{b_1} p_2^{b_2} p_3^{b_3} \cdots \quad / b_i \ge 0 \text{ for } i=1,2,3,\ldots$$

$$c = p_1^{c_1} p_2^{c_2} p_3^{c_3} \cdots \quad / c_i \ge 0 \text{ for } i=1,2,3,\ldots$$

From (A7.1.2), since $(a,b) = 1 = (a,c)$, it follows that:

$\mathrm{MIN}\{a_i,b_i\} = \mathrm{MIN}\{a_i,c_i\} = 0 \quad /i=1,2,3,\ldots \implies$

Either $a_i = 0$ or $b_i = c_i = 0 \quad /i=1,2,3,\ldots \implies$

Either $a_i = 0$ or $b_i + c_i = 0 \quad /i=1,2,3,\ldots \implies$

$\mathrm{MIN}\{a_i,b_i+c_i\} = 0 \quad /i=1,2,3,\ldots \implies$

$(a,bc) = 1$ [by (A7.1.2)]

QED

<u>Theorem A7.1.14</u>: For any a, b & c, such that $(a,b) = 1$:

$$a \mid c \;\; \& \;\; b \mid c \implies ab \mid c \qquad \text{(A7.1.9)}$$

<u>Proof</u>: Since $a|c$ & $b|c$, there exist integers q & s, such that $c = qa = sb \implies b|qa$. Since $(a,b) = 1 \implies b|q$ (by Theorem A7.1.10), hence there exists integer t, such that $q=tb$. Then, $c=tba \implies ab|c$.

QED

<u>Theorem A7.1.15</u>: If $m>1$ has the prime decomposition:

$$m = p_1^{a_1} p_2^{a_2} \cdots p_r^{a_r} \quad / p_1<p_2<\cdots<p_r \;\&\; a_i \ge 1 \text{ for } i=1,2,\ldots,r$$

Then:

$$\Phi(m) = m\Big[(p_1-1)(p_2-1)\cdots(p_r-1)\Big]\Big/\Big[p_1 p_2 \cdots p_r\Big] \qquad \text{(A7.1.10)}$$

<u>Proof</u>: See Apostol [44], p. 27.

∎

APPENDIX 7.2 INTRODUCTION TO CONGRUENCES

This appendix, like Appendix 7.1, is based on Apostol's textbook "Introduction to Analytic Number Theory" [44]. The material (definitions & theorems) has been drawn mainly from

Chapter 5.

Unless otherwise stated, small latin & greek letters will denote integers.

<u>Definition A7.2.1</u>: Given α,b & m, with m>0, it is said that α is *congruent* to b *modulo* m, denoted by $\alpha \equiv b \pmod{m}$, if m divides the difference α-b. m is called the *modulus* of the congruence:

$$\alpha \equiv b \pmod{m} \quad \Longleftrightarrow \quad m \mid (\alpha-b) \qquad (A7.2.1)$$

∎

<u>Theorem A7.2.1</u>: Congruence is an equivalence relation; in other words it is reflective, symmetric and transitive. For any α, b, c & m, with m>0:

$$\alpha \equiv \alpha \pmod{m} \qquad (A7.2.2a)$$

$$\alpha \equiv b \pmod{m} \quad \Longrightarrow \quad b \equiv \alpha \pmod{m} \qquad (A7.2.2b)$$

$$\left.\begin{array}{l} \alpha \equiv b \pmod{m} \\ b \equiv c \pmod{m} \end{array}\right] \Longrightarrow \quad \alpha \equiv c \pmod{m} \qquad (A7.2.2c)$$

<u>Proof</u>: See Apostol [44], p. 107.

∎

<u>Theorem A7.2.2</u>: For any α, b, c, d & m, with m>0, if $\alpha \equiv b \pmod{m}$ & $c \equiv d \pmod{m}$, then:

$$\alpha c \equiv bd \pmod{m} \qquad (A7.2.3a)$$

For all integers x & y, $\alpha x+cy \equiv bx+dy \pmod{m}$ (A7.2.3b)

For all positive integers n, $\alpha^n \equiv b^n \pmod{m}$ (A7.2.3c)

For every polynomial f with integer coefficients,

$$f(\alpha) \equiv f(b) \pmod{m} \qquad (A7.2.3d)$$

<u>Proof</u>: See Apostol [44], p. 107.

∎

<u>Theorem A7.2.3</u>: For any α, b & m, such that $0 \le |\alpha-b| < m$:

$$\alpha = b \quad \Longleftrightarrow \quad \alpha \equiv b \pmod{m} \qquad (A7.2.4)$$

<u>Proof</u>: Obviously, if $\alpha = b \Longrightarrow \alpha \equiv b \pmod{m}$. Let $\alpha \equiv b$

(mod m); from (A7.2.1), $m|(a-b) \longrightarrow a-b = km$. On the other hand, by hypothesis, $0 \le |a-b| < m$; hence $0 \le |k|m < m \longrightarrow 0 \le |k| < 1 \longrightarrow k=0 \longrightarrow a=b$.

QED

Theorem A7.2.4: For any a, b, c & m, with m>0, if $ac \equiv bc \pmod m$ and if $d \triangleq (m,c)$, then $a \equiv b \pmod{m/d}$.

Proof: See Apostol [44], p. 109.

∎

Definition A7.2.2: The set of all integers x such that $x \equiv a \pmod m$, where m>0, is called the *residue class* a modulo m. A set of m representatives, one from each of the residue classes a modulo m /a=0,1,...,m-1, is called a *complete residue system* modulo m. Hence, {0,1,2,...,m-1}, {1,2,3,...,m}, etc, are complete residue systems modulo m.

∎

Theorem A7.2.5: Assume $(a,m) \triangleq d$. Then, the linear congruence $ax \equiv b \pmod m$ has solutions if, and only if, d|b. Furthermore, and if d|b, the congruence has exactly d solutions modulo m, given by t+im/d /i=0,1,..,d-1, where t is the solution, unique modulo m/d, of the congruence $a/d \equiv b/d \pmod{m/d}$. It is understood that 'a solution of a congruence modulo m' means a number within a complete residue system modulo m, say {0,1,..,m-1}, satisfying that congruence.

Proof: See Apostol [44], pp. 111-2.

∎

Definition A7.2.3: Any set of $\Phi(m)$ integers, incongruent modulo m, each of which is relatively prime to m, is called a *reduced residue system* modulo m.

∎

Theorem A7.2.6: *Euler-Fermat theorem*: For any a & m, with m>0, if (a,m)=1 then

$$a^{\Phi(m)} \equiv 1 \pmod m \qquad (A7.2.5)$$

Proof: See Apostol [44], p. 113.

∎

Theorem A7.2.7: For any a, b & m, with $m>0$, if $(a,m)=1$ then the solution (unique modulo m) of the linear congruence $ax \equiv b \pmod{m}$ is given by

$$x \equiv ba^{\phi(m)-1} \pmod{m} \qquad \text{(A7.2.6)}$$

Proof: See Apostol [44], p. 114.

∎

Theorem A7.2.8: For any a, b & m, with $m>0$ and $a \equiv b \pmod{m}$, if $d|m$ and $d|a$, then $d|b$.

Proof: See Apostol [44], p. 109.

∎

Theorem A7.2.9: For any a & m, with $m>0$:

$$(a,m) > 1 \Longrightarrow a^n \not\equiv 1 \pmod{m} \quad /n=1,2,\ldots \qquad \text{(A7.2.7a)}$$

$$(a,m) = 1 \Longrightarrow a^{\phi(m)} \equiv 1 \pmod{m} \qquad \text{(A7.2.7b)}$$

(A7.2.7b) is known as the *Euler-Fermat Theorem.* *

Proof: (A7.2.7b) is Theorem A7.2.6, included here to complete the case.

Let $(a,m) \hat{=} d > 1$ and assume that there exist $k>1$ such that $a^k \equiv 1 \pmod{m}$. Since $d|a \Longrightarrow d|a^k$. Also, $d|m$. Then, by Theorem A7.2.8, $d|1 \Longrightarrow d=1 \Longrightarrow$ contradiction.

QED

Theorem A7.2.10: The *Chinese remainder theorem*: Assume $m_1,m_2,\ldots,m_r$ are relatively prime in pairs. Let $b_1,b_2,\ldots,b_r$ be arbitrary integers and let $a_1,a_2,\ldots,a_r$ satisfy $(a_i,m_i) = 1$ $/i=1,2,\ldots,r$. Then the linear system of congruences $a_ix_i \equiv b_i \pmod{m_i}$ $/i=1,2,\ldots,r$ has exactly one solution modulo $m_1m_2\cdots m_r$.

Proof: See Apostol [44], p. 118.

∎

Theorem A7.2.11: For any a, b & m, with $m>0$:

$$a \equiv b \pmod{m} \Longrightarrow (a,m) = (b,m) \qquad \text{(A7.2.8)}$$

Proof: See Apostol [44], p. 109.

∎

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

APPENDIX 7.3: INTRODUCTION TO PRIMITIVE ROOTS

This appendix is drawn mainly from Chapter 10 of T.M. Apostol's "Introduction to Analytic Number Theory" ([44]).

Unless otherwise stated, small latin & greek letters will denote integers.

Definition A7.3.1: Let α & m, with m>0. The smallest positive integer f, such that:

$$\alpha^f \equiv 1 \pmod{m}$$

is called the *order* (or *exponent*) of α modulo m and is denoted by $\mathrm{Ord}_m(\alpha)$ $[\exp_m(\alpha)]$. If $\mathrm{Ord}_m(\alpha) = \Phi(m)$, then α is called a *primitive root* modulo m. [44]

∎

Theorem A7.3.1: For any α, m & k, with m & k positive:

$$\mathrm{Ord}_m(\alpha^k) = \mathrm{Ord}_m(\alpha) \Big/ \left(\mathrm{Ord}_m(\alpha), k\right) \qquad \text{(A7.3.1)}$$

Proof: See Apostol [44], p. 206.

∎

Theorem A7.3.2: For any k, n & m positive and any α, if $\mathrm{Ord}_m(\alpha) \triangleq f$, then:

i) $\alpha^k \equiv \alpha^n \pmod{m} \iff k \equiv n \pmod{f}$ (A7.3.2a)

ii) $\alpha^k \equiv 1 \pmod{m} \iff k \equiv 0 \pmod{f}$ (A7.3.2b)

iii) $f \mid \Phi(m)$ (A7.3.2c)

iv) The numbers $1, \alpha, \alpha^2, \ldots, \alpha^{f-1}$ are incongruent (mod m).

Proof: See Apostol [44], p. 205.

∎

Theorem A7.3.3: Let p be any odd prime and S any positive divisor of p-1. Then in every reduced residue system modulo p there are exactly $\Phi(S)$ numbers α such that $\mathrm{Ord}_p(\alpha) = S$.

Proof: See Apostol [44], pp. 207-8.

∎

Theorem A7.3.4: Let p be an odd prime. Then, if g is a primitive root modulo p, g is also a primitive root modulo p^{α} for all $\alpha \geq 1$ if, and only if:

$$g^{p-1} \not\equiv 1 \pmod{p^2} \qquad \text{(A7.3.3)}$$

Furthermore, there is at least one primitive root g modulo p which satisfies (A7.3.3).

Proof: See Apostol [44], pp. 209-10.

∎

Theorem A7.3.5: Let any odd prime p and a positive divisor S of p-1. If g is a primitive root modulo p, satisfying:

$$g^{p-1} \not\equiv 1 \pmod{p^2} \qquad \text{(A7.3.3)}$$

then, for any $\alpha \geq 1$, $g^{\Phi(m)/S}$ (where $m=p^{\alpha}$) has order S modulo p^b, for any $b=1,2,\ldots,\alpha$.

Proof: Theorem A7.3.3 guarantees the existence of $\Phi(p-1)$ primitive roots modulo p. Furthermore, if one of them, say g, satisfies (A7.3.3), then g is also a primitive root modulo p^{α}, for any $\alpha \geq 1$, by Theorem A7.3.4. According to the same theorem, there is at least one primitive root of p which satisfies (A7.3.3). Let g be the one. Then:

$$\mathrm{Ord}_n(g) = \Phi(n) \quad /n=p^b \;\&\; b\geq 1 \qquad \text{(A)}$$

From Theorem A7.3.1 & (A):

$$\mathrm{Ord}_n\left(g^{\Phi(m)/S}\right) = \Phi(n) \Big/ \left(\Phi(n),\Phi(m)/S\right) \quad /m=p^{\alpha} \qquad \text{(B)}$$

Let $$f \triangleq \left(\Phi(p^b),\Phi(p^{\alpha})/S\right)^{*} \qquad \text{(C)}$$

Using Theorem A7.1.15:

$$f = \left(p^{b-1}(p-1),p^{\alpha-1}(p-1)/S\right) \longrightarrow$$

$$f = \left(p^{b-1}(p-1)/S\right)\left(S,p^{\alpha-b}\right) \qquad \text{(D)}$$

using (A7.1.1c), the hypothesis that $b\leq\alpha$ and the fact that (p-1)/S divides p-1.

Since $S \mid p-1 \longrightarrow S < p \longrightarrow (S,p) = 1$ (by Theorem A7.1.9) $\longrightarrow (S,p^{\alpha-b}) = 1$ (by Theorem A7.1.11). Then from (D): $f = p^{b-1}(p-1)/S$. Then, from (B):

* (a,b) denotes the *greatest common divisor* of a & b.

$$\mathrm{Ord}_n\left(g^{\phi(m)/S}\right) = p^{b-1}(p-1)\Big/\left[p^{b-1}(p-1)/S\right] = S$$

QED

<u>Note A7.3.1</u>: TABLE A7.3.1 below, lists the smallest primitive root modulo p, for all integers n < 607 that have a primitive root. The roots were calculated using *subroutine IPRIM1* (for a flow-chart of IPRIM1 see Fig. A8.1.6, p. 515).

<u>TABLE A7.3.1</u>

n	g	n	g	n	g	n	g	n	g	n	g	n	g
1	1	47	5	118	11	199	3	289	3	386	5	491	2
2	1	49	3	121	2	202	3	293	2	389	2	499	7
3	2	50	3	122	7	206	5	298	3	394	3	502	11
4	3	53	2	125	2	211	2	302	7	397	5	503	5
5	2	54	5	127	3	214	5	307	5	398	3	509	2
6	5	58	3	131	2	218	11	311	17	401	3	514	3
7	3	59	2	134	7	223	3	313	10	409	21	521	3
9	2	61	2	137	3	226	3	314	5	419	2	523	2
10	3	62	3	139	2	227	2	317	2	421	2	526	5
11	2	67	2	142	7	229	6	326	3	422	3	529	5
13	2	71	7	146	5	233	3	331	3	431	7	538	3
14	3	73	5	149	2	239	7	334	5	433	5	541	2
17	3	74	5	151	6	241	7	337	10	439	15	542	15
18	5	79	3	157	5	242	7	338	7	443	2	547	2
19	2	81	2	158	3	243	2	343	3	446	3	554	5
22	7	82	7	162	5	250	3	346	3	449	3	557	2
23	5	83	2	163	2	251	6	347	2	454	5	562	3
25	2	86	3	166	5	254	3	349	2	457	13	563	2
26	7	89	3	167	5	257	3	353	3	458	7	566	3
27	2	94	5	169	2	262	17	358	7	461	2	569	3
29	2	97	5	173	2	263	5	359	7	463	3	571	3
31	3	98	3	178	3	269	2	361	2	466	3	577	5
34	3	101	2	179	2	271	6	362	21	467	2	578	3
37	2	103	5	181	2	274	3	367	6	478	7	586	3
38	3	106	3	191	19	277	5	373	2	479	13	587	2
41	6	107	2	193	5	278	3	379	2	482	7	593	3
43	3	109	6	194	5	281	3	382	19	486	5	599	7
46	5	113	3	197	2	283	3	383	5	487	3	601	7

APPENDIX 7.4: PROOF OF THEOREM 7.2

A code is not self-orthogonal, if two syndrome bits that check on the same error bit, say $e_0^{(i)}$, check also on another error bit. Consider two such syndromes, say $s_u^{(r)}$ & $s_w^{(v)}$.

From (7.5) (p. 183), the syndrome eqns are:

$$s_u^{(r)} = e_0^{\{a[r,u+1]\}} + e_1^{\{a[r,u]\}} +\dots+ e_c^{\{a[r,u+1-c]\}} +\dots+ e_u^{\{a[r,1]\}} + e_u^{(k+r)}$$

$$s_w^{(v)} = e_0^{\{a[v,w+1]\}} + e_1^{\{a[v,w]\}} +\dots+ e_c^{\{a[v,w+1-c]\}} +\dots+ e_w^{\{a[v,1]\}} + e_w^{(k+v)}$$

Using the fact that they both check the ith bit of $(e^{*})_0$, i.e. that

$$a_{r,u+1} = a_{v,w+1} = i \qquad (A)$$

$$s_u^{(r)} = e_0^{(i)} + e_1^{\{a[r,u]\}} +\dots+ e_c^{\{a[r,u+1-c]\}} +\dots+ e_u^{\{a[r,1]\}} + e_u^{(k+r)} \qquad (B)$$

$$s_w^{(v)} = e_0^{(i)} + e_1^{\{a[v,w]\}} +\dots+ e_c^{\{a[v,w+1-c]\}} +\dots+ e_w^{\{a[v,1]\}} + e_w^{(k+v)} \qquad (C)$$

As 'promised' earlier on, let these two syndromes check also on <u>another</u> common error bit, say $e_c^{(b)}$ /c>0. Then, the corresponding IA elements will both be equal to b. From eqns (B) & (C), the coefficient of the error bit from the cth block, participating in the formation of $s_u^{(r)}$, is $a_{r,u+1-c}$ (=b) and the coefficient of the error bit from the cth block, participating in the formation of $s_w^{(v)}$, is $a_{v,w+1-c}$ (=b).

Since $a_{r,u+1}$, $a_{v,w+1}$, $a_{r,u+1-c}$ & $a_{v,w+1-c}$, are elements of an (n-k) x (m+1) array of integers, $1\le r,v\le n-k$ and $1\le u+1,w+1,u+1-c,w+1-c\le m+1$. The second inequality gives $0\le u,w\le m$ and $0\le u-c\le m$ & $0\le w-c\le m$. The latter is equivalent to

$$(-m\le c-u\le 0 \ \& \ -m\le c-w\le 0) \iff (u-m\le c\le u \ \& \ w-m\le c\le w)$$

and since u-m & w-m are at most 0, while c has to be positive [see (B) or (C)], $0<c\le MIN\{u,w\}$. Note also that if u=0, $s_u^{(r)}$ cannot check on $e_c^{(b)}$ /c>0, as well [see eqn (B)]; hence $u\ne 0$. Similarly, $w\ne 0$.

Hence, if the code generated by the IA is not self-orthogonal, there exist numbers r,u,v,w & c, such that:

$$1\le r,v\le n-k \quad \& \quad 1\le u,w\le m \quad \& \quad 0<c\le MIN\{u,w\} \qquad (D)$$

and $$a_{r,u+1} = a_{v,w+1} \quad \& \quad a_{r,u+1-c} = a_{v,w+1-c} \qquad (E)$$

Conversely, assume that there exist numbers r,u,v,w & c such that (D) & (E), above, hold true. Then, it is noted from the first of (E) that, syndrome bits $s_u^{(r)}$ and $s_w^{(v)}$ both check on error bit $e_0^{(i)}$ [where $i = a_{r,u+1} = a_{v,w+1}$], but because of the second of (E) they both check on $e_c^{(b)}$ [where $b = a_{r,u+1-c} = a_{v,w+1-c}$] and because $c>0$, the corresponding code is not self-orthogonal.

Note also that c is the positional difference between any two distinct elements of a row and since the IA has m+1 columns, c ranges between 1 & $MIN\{u,w\} \le m$.

If u+1 & w+1, in eqns (D) & (E), are replaced by u & w, the theorem is proved.

QED

APPENDIX 7.5: THE MINIMUM VALUE OF m FOR TYPE-B CODES

A7.5.1. Proof of Theorem 7.3

The elements of the IA denote the position of a message bit within a block, hence $1 \le a_{x,z} \le k$. Reln (7.8) restricts the elements in the range [0,k], hence the generation of elements along a row must stop just before the generation of the first 0. Then the smallest integer z, in the range [1,k] satisfying the linear congruence $za_{x,1} \equiv 0 \pmod{k+1}$, will give the position of the first zero along row x.

According to Theorem A7.2.5 the above linear congruence has exactly d solutions, where $d \triangleq (k+1,a_{x,1})$ (because $d|0$). The solutions are given by $t+i(k+1)/d$ /$i=0,1,..,d-1$, where t is the solution of $t(a_{x,1}/d) \equiv 0 \pmod{(k+1)/d}$; according to Theorem A7.2.7, $t \equiv 0 \pmod{(k+1)/d}$, hence $z \equiv i(k+1)/d \pmod{k+1}$ /$i=0,1,..,d-1$. Hence the 1st zero along row x is at position $(k+1)/(k+1,a_{x,1})$ and so the number of elements of row x must be $(k+1)/(k+1,a_{x,1})-1$. In general, the length of the rows of the IA will vary, between 1 & k. This means that the condition introduced by Definition 7.1 will not be satisfied, in general. Definition 7.1 requires all rows to have the same length, which by necessity will be the length of the shortest row.

This proves the theorem.

QED

A7.5.2. Proof of Theorem 7.4

From Theorem 7.3, in order that the IA contains no entries equal to zero, it is necessary that the maximum value of m, m_{max}, satisfies

$$m_{max} = \underset{x=1}{\overset{n-k}{\mathrm{MIN}}}\left\{(k+1)/(k+1,a_{x,1})\right\} - 2 \qquad \text{(A)}$$

where $a_{x,1}$ /x=1,2,...,n-k are the elements of the 1st column of the corresponding IA. Let $i \triangleq (k+1,a_{x,1}) \Longleftrightarrow i(k+1) = (k+1,a_{x,1})(k+1) \Longleftrightarrow i(k+1)/(k+1,a_{x,1}) = k+1$. Then k+1 is divided by $(k+1)/(k+1,a_{x,1})$ and hence $1 \le (k+1)/(k+1,a_{x,1}) \le k+1$. Assume that $(k+1)/(k+1,a_{x,1}) = 1$; then $(k+1,a_{x,1}) = k+1 \Longrightarrow (k+1)|a_{x,1} \Longrightarrow a_{x,1} \ge k+1$. On the other hand, by construction, $1 \le a_{x,1} \le k$. Hence contradiction* and $(k+1)/(k+1,a_{x,1}) \ne 1$. Hence, $(k+1)/(k+1,a_{x,1})$ is a divisor of k+1, which is not 1, i.e. which is greater than 1.

Then, the minimum of $(k+1)/(k+1,a_{x,1})$ /x=1,2,...,n-k is the minimum of a set of non-trivial (i.e. different than one) divisors of k+1 and obviously it cannot be smaller than the minimum non-trivial divisor of k+1; the latter may only be a prime, say, p (because, if not, there will be a prime diving it and, hence, k+1 as well). Hence, the right-hand side of eqn (A) is $\ge$ p-2.

For (2k,k,m) codes, the first column, i.e. elements $a_{x,1}$ /x=1,2,...,n-k, contains k distinct (mod k+1) elements, in the range [1,k]. Note that, if any two elements, $a_{r,1} = a_{v,1}$ /$r \ne v$, are equal, the code will not be self-orthogonal, according to Theorem 7.2, because rows r & v will be identical, given the IA construction-technique introduced by Definition 7.2. Hence, $a_{x,1}$ 'runs' through the range [1,k]. According to Theorem A7.1.7, if $a_{x,1}$ 'runs' through the range [1,k+1], then $(k+1)/(k+1,a_{x,1})$ 'runs' through the set of positive divisors of k+1. Since $1 \le a_{x,1} \le k$, divisor $(k+1)/(k+1,k+1)=1$ is excluded and, since no $a_{x,1}<k+1$ can generate $(k+1)/(k+1,a_{x,1})=1$, $(k+1)/(k+1,a_{x,1})$ 'runs' through the

* Anyway, if $(k+1)/(k+1,a_{x,1})=1$, then $m_{max}=-1$, hence there is no IA, or code.

set of divisors of k+1 that are greater than 1. Hence, p is definitely equal to one of $(k+1)/(k+1,a_{x,1})$ $/1\le x\le n-k$ and m_{max} = p-2.

QED

APPENDIX 7.6: ORTHOGONALITY CONDITIONS FOR TYPE-B CODES

A7.6.1. Proof of Lemma 7.1

Consider two elements, along row x of the IA, say elements $a_{x,z+c}$ & $a_{x,z}$, where z & c are positive integers. Then, from (7.8) (p. 185) and since, by Theorem A7.2.2, congruences may be added, subtracted or multiplied member by member as though they were equations:

$$a_{x,z+c} - a_{x,z} \equiv (z+c)a_{x,1} - za_{x,1} \pmod{k+1}$$

$$\Longrightarrow \quad a_{x,z+c} - a_{x,z} \equiv ca_{x,1} \pmod{k+1}$$

QED

A7.6.2. Proof of Theorem 7.5

According to Theorem 7.2, the code is <u>not</u> self-orthogonal if, and only if, there is at least one pair of elements $a_{r,u} = a_{v,w}$ and at least one integer c, such that $a_{r,u-c} = a_{v,w-c}$, where 0<c<MIN{u,w}. Note though that:

$$\begin{array}{l} a_{r,u} = a_{v,w} \\ a_{r,u-c} = a_{v,w-c} \end{array} \Longleftrightarrow \begin{array}{l} a_{r,u} = a_{v,w} \\ a_{r,u-c} - a_{r,u} = a_{v,w-c} - a_{v,w} \end{array}$$

$$\Longleftrightarrow \begin{array}{l} a_{r,u} \equiv a_{v,w} \pmod{k+1} \\ a_{r,u-c} - a_{r,u} \equiv a_{v,w-c} - a_{v,w} \pmod{k+1} \end{array}$$

The last result is obtained from Theorem A7.2.3, noting that the elements of the IA are always in the range [1,k], hence the absolute value of the difference of any two of them is less than k. Using (7.11), the last result gives:

$$\begin{array}{l} a_{r,u} = a_{v,w} \\ a_{r,u-c} = a_{v,w-c} \end{array} \Longleftrightarrow \begin{array}{l} a_{r,u} = a_{v,w} \\ c(a_{v,1} - a_{r,1}) \equiv 0 \pmod{k+1} \end{array}$$

Hence, a type-B code is not self-orthogonal if, and only if, for any two rows, say, r & v which have a common element, in columns u & w, there exists at least one positive integer c, less than u & w, such that $ca_{r,1}$ is congruent to $ca_{v,1}$ modulo k+1. Condition (7.9) is a necessary restriction on m, imposed by the introduction of the generation method of Definition 7.2.

QED

APPENDIX 7.7: PROPERTIES OF TYPE-B CODES

A7.7.1. Proof of Theorem 7.6

Assume that there exists a row, say, x ($1 \le x \le n-k$) which has at least one pair of equal elements, say $a_{x,u}=a_{x,w}$ ($1 \le u \ne w \le m+1$). Then, from Definition 7.2 & Theorem A7.2.4:

$$ua_{x,1} \equiv wa_{x,1} \pmod{k+1} \Longrightarrow$$

$$u \equiv w \pmod{(k+1)/(k+1,a_{x,1})}^{*} \qquad \text{(A)}$$

Note though that, from relation (7.9), $u,w \le m+1 \le (k+1)/(k+1,a_{x,1})-1 \Longrightarrow 0<|u-w|<(k+1)/(k+1,a_{x,1})-1$ and then, by Theorem A7.2.3 & (A), u=w which contradicts the hypothesis. Hence, the first of the two results.

Consider now any specific column, say, u ($1 \le u \le m+1$) and let two of its elements, say $a_{r,u}=a_{v,u}$ ($1 \le r \ne v \le n-k$), be equal. According to Definition 7.2 and Theorem A7.2.4:

$$ua_{r,1} \equiv ua_{v,1} \pmod{k+1} \Longrightarrow$$

$$\Longrightarrow a_{r,1} \equiv a_{v,1} \pmod{(k+1)/(k+1,u)} \qquad \text{(B)}$$

Conversely, let (B) hold true. From Definition A7.2.1:

$$(k+1)/(k+1,u) \text{ divides } (a_{r,1}-a_{v,1})$$

$$\Longrightarrow a_{r,1}-a_{v,1} = q(k+1)/(k+1,u)$$

$$\Longrightarrow u(a_{r,1}-a_{v,1}) = q[u/(k+1,u)](k+1)$$

$$\Longrightarrow (k+1) \mid (ua_{r,1}-ua_{v,1})$$

$$\Longrightarrow ua_{r,1} \equiv ua_{v,1} \pmod{k+1} \Longrightarrow a_{r,u} \equiv a_{v,u} \pmod{k+1}$$

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

From Theorem A7.2.3, and since $1 \le a_{r,u}, a_{v,u} \le k \implies 0 \le |a_{r,u}-a_{v,u}| < k$, $a_{r,u}=a_{v,u}$.

QED

A7.7.2. Proof of Theorem 7.7

Let the elements of the first column, $a_{x,1}$ /x=1,2,...,n-k, be distinct (obviously in the range [1,k]). Then for any two rows, say, $r \ne v$, $a_{r,1} \ne a_{v,1}$ and since $1 \le a_{r,1}, a_{v,1} \le k$, it follows that $0 < |a_{r,1}-a_{v,1}| < k$, $a_{r,1} \not\equiv a_{v,1}$ (mod k+1), by Theorem A7.2.3. Let c be any integer in the range [1,m+1]. Since $c \le m+1 < p$, $(c,k+1)=1$, by Theorem A7.1.8.

So, $a_{r,1} \not\equiv a_{v,1}$ (mod (k+1)/(k+1,c)) for all $r \ne v$ and all c=1,2,...,m+1. Then, by the corollary of Theorem 7.5, the code is self-orthogonal.

Conversely, let the code be self-orthogonal and <u>assume</u> that there exist two elements in the first column that are equal, say elements $a_{r,1}=a_{v,1}$. Then they are congruent modulo anything, hence relation (7.13b) does not hold true and the code is not self-orthogonal. This <u>contradicts</u> the initial hypothesis, hence there are no equal elements in the first column.

QED

A7.7.3. Proof of Theorem 7.8

According to Theorem 7.7, the first column contains n-k distinct integers in the range [1,k]. Clearly, $n-k \le k \implies 1-R \le R \implies R \ge 1/2$.

<u>Assume</u> k=odd. Then k+1=even and $p=2 \implies m \le p-2=0 \implies m=0$, hence the code is not (even) convolutional, hence <u>contradiction</u>. Then k=even.

<u>Assume</u> that there exists at least one column, say, u ($1<u \le m+1$) with at least two equal elements, say $a_{r,u}=a_{v,u}$ ($1 \le r \ne v \le n-k$). Then, by Theorem 7.6, $a_{r,1} \equiv a_{v,1}$ (mod (k+1)/(k+1,u)). Since $1<u \le m+1<p$, $(k+1,u)=1$, by Theorem A7.1.8, and $a_{r,1} \equiv a_{v,1}$ (mod k+1). Because $a_{r,1}$ & $a_{v,1}$ are generated modulo k+1, they are equal. But this is equivalent to the code not being self-orthogonal (according to Theorem

7.7), which contradicts the hypothesis. Hence each column contains a distinct set of integers.

QED

A7.7.4. Proof of Theorem 7.9

According to Definition 5.5, the effective constraint-length, n_E, for the decoding of $e_h^{(i)}$ is equal to the sum of the sizes of the composite parity checks (in this case, the syndrome bits), that are orthogonal on $e_h^{(i)}$, plus one. Also, according to Definition 5.4, if a code is self-orthogonal, all the syndrome bits checking on $e_h^{(i)}$ are orthogonal on it. Furthermore, the decoding circuit for $e_h^{(i)}$ is identical to that for $e_0^{(i)}$ [see discussion following equations (5.14), p. 141]. Hence, n_E-1 [for $e_h^{(i)}$] equals the sum of the sizes of the, say, J syndromes checking on $e_0^{(i)}$.

According to Theorem 7.1, $e_0^{(i)}$ is checked by syndrome bit $s_{w-1}^{(x)}$ if, and only if, $a_{x,w} = i$. So, the number of syndromes, checking on $e_0^{(i)}$, equals the number of IA entries that are equal to i and, as a consequence, this number is equal to J. Then, according to the above discussion, there will exist J IA elements equal to i:

$$a[x_1,w_1] = a[x_2,w_2] = \cdots = a[x_J,w_J] = i \qquad \text{(A)}$$

Then, the J syndromes checking on $e_0^{(i)}$ are $s_{w(j)-1}^{\{x(j)\}}$ for j=1,2,...,J [where $x(j) \triangleq x_j$ & $w(j) \triangleq w_j$] and, according to eqn (7.6):

$$s_{w(j)-1}^{\{x(j)\}} = \sum_{z=1}^{w(j)} e_{w(j)-z}^{\{a[x(j),z]\}} + e_{w(j)-1}^{\{k+x(j)\}} \qquad \text{(B)}$$

Expanding eqns (B):

$$s_{w(1)-1}^{\{x(1)\}} = e_0^{(i)} + e_1^{\{a[x(1),w(1)-1]\}} + \cdots + e_{w(1)-1}^{\{a[x(1),1]\}} + e_{w(1)-1}^{\{k+x(1)\}}$$

$$s_{w(2)-1}^{\{x(2)\}} = e_0^{(i)} + e_1^{\{a[x(2),w(2)-1]\}} + \cdots + e_{w(2)-1}^{\{a[x(2),1]\}} + e_{w(2)-1}^{\{k+x(2)\}}$$

. . . .

$$s_{w(J)-1}^{\{x(J)\}} = e_0^{(i)} + e_1^{\{a[x(J),w(J)-1]\}} + \cdots + e_{w(J)-1}^{\{a[x(J),1]\}} + e_{w(J)-1}^{\{k+x(J)\}}$$

The size* of the above syndromes is $w_1,w_2,\ldots,w_J$, respectively. Then, by eqn (5.9),

* The size excludes the checked bit - see Definition 5.5.

$$n_E = 1 + w_1 + w_2 + \cdots + w_J \quad (C)$$

On the other hand, according to Definition 7.3 and taking into account (A), above, the leftwise sequences on $e_h^{(i)}$ are

$$a[x_j,w_j]\ a[x_j,w_{j-1}]\ a[x_j,w_{j-2}]\ \cdots\ a[x_j,w_2]\ a[x_j,w_1]\ /\ j=1,2,..,J$$

hence, the number of elements in the leftwise sequences is $w_1 + w_2 + \cdots + w_J$, which equals to n_E-1, according to eqn (C).

QED

APPENDIX 7.8: TYPE-B1 CODES

A7.8.1. Examples

Example A7.8.1: Let the initial array for the (14,J) type-B1 code. Since k+1=15, then p=3 and $2 \le J \le p-1=2 \Longrightarrow J=2$. Hence, the IA is an $(n-k) \times (m+1) = k \times J = 14 \times 2$ array.

As predicted by Theorem 7.10, there are exactly J=2 syndromes checking on each error bit. Hence, the above is a (28,14,1) systematic CSOC which can correct up to one error within one constraint-length $[n_A = n(m+1) = 28 \times 2 = 56]$.

1	2
2	4
3	6
4	8
5	10
6	12
7	14
8	1
9	3
10	5
11	7
12	9
13	11
14	13

∎

Example A7.8.2: Let the initial array for the (24,J) type-B1 code. Since k+1=25, p=5 and $2 \le J \le p-1=4$. Let J=4. Then the IA is a 24 x 4 array:

1	2	3	4
2	4	6	8
3	6	9	12
4	8	12	16
5	10	15	20
6	12	18	24
7	14	21	3
8	16	24	7
9	18	2	11
10	20	5	15
11	22	8	19
12	24	11	23
13	1	14	2
14	3	17	6
15	5	20	10
16	7	23	14
17	9	1	18
18	11	4	22
19	13	7	1
20	15	10	5
21	17	13	9
22	19	16	13
23	21	19	17
24	23	22	21

The leftwise sequences, for selected error bits, are given below:

1				2				5				24			
1	13			2	1			5	15			24	12		
1	9	17		2	18	9		5	20	10		24	16	8	
1	7	13	19	2	14	1	13	5	10	15	20	24	18	12	6

If all leftwise sequences are checked, it will be verified that the associated code is indeed a J=4 (48,24,3) systematic CSOC which can correct up to 2 errors in one constraint-length [n_A = 48 x 4 = 192].

■

<u>Example A7.8.3</u>: Consider the (48,24,3) code of Example A7.8.2 and the decoding of the, say, 21st current message bit, $r_h^{(21)}$ [or, the same, the estimation of $e_h^{(21)}$]. The 4 syndrome eqns that contain the 21st current message error bit may be deduced from the leftwise sequences for (21).

$$s_g^{(7)} = e_g^{(7)} + e_{g-1}^{(14)} + e_{g-2}^{(21)} + e_{g-3}^{(3)} + e_g^{(31)}$$

$$s_g^{(21)} = e_g^{(21)} + e_{g-1}^{(17)} + e_{g-2}^{(13)} + e_{g-3}^{(9)} + e_g^{(45)}$$

$$s_g^{(23)} = e_g^{(23)} + e_{g-1}^{(21)} + e_{g-2}^{(19)} + e_{g-3}^{(17)} + e_g^{(47)}$$

$$s_g^{(24)} = e_g^{(24)} + e_{g-1}^{(23)} + e_{g-2}^{(22)} + e_{g-3}^{(21)} + e_g^{(48)}$$

From the above four equations, it is obvious that the four syndromes checking on $e_h^{(21)}$ are described by the following equations.

$$
\begin{aligned}
s_{h+2}^{(7)} &= e_{h+2}^{(7)} + e_{h+1}^{(14)} + e_{h}^{(21)} + e_{h-1}^{(3)} + e_{h+2}^{(31)} \\
s_{h}^{(21)} &= e_{h}^{(21)} + e_{h-1}^{(17)} + e_{h-2}^{(13)} + e_{h-3}^{(9)} + e_{h}^{(45)} \\
s_{h+1}^{(23)} &= e_{h+1}^{(23)} + e_{h}^{(21)} + e_{h-1}^{(19)} + e_{h-2}^{(17)} + e_{h+1}^{(47)} \\
s_{h+3}^{(24)} &= e_{h+3}^{(24)} + e_{h+2}^{(23)} + e_{h+1}^{(22)} + e_{h}^{(21)} + e_{h+3}^{(48)}
\end{aligned}
$$

Assuming feedback decoding and no past errors (or 'genie decoding' - see Chapter 6, p. 157), the past error bits, i.e. $e_g^{(i)}$ $/g<h$, are correctly estimated and cancelled out. Then, the above four equations are modified to:

$$
\begin{aligned}
s_{h}^{(21)} &= e_{h}^{(21)} + e_{h}^{(45)} \\
s_{h+1}^{(23)} &= e_{h+1}^{(23)} + e_{h}^{(21)} + e_{h+1}^{(47)} \\
s_{h+2}^{(7)} &= e_{h+2}^{(7)} + e_{h+1}^{(14)} + e_{h}^{(21)} + e_{h+2}^{(31)} \\
s_{h+3}^{(24)} &= e_{h+3}^{(24)} + e_{h+2}^{(23)} + e_{h+1}^{(22)} + e_{h}^{(21)} + e_{h+3}^{(48)}
\end{aligned}
$$

It is obvious that $e_h^{(21)}$ will be correctly calculated, using the majority-decoding algorithm (Theorem 5.3), if no more than two of the 11 bits appearing in the above four equations have been corrupted. Hence, up to 2 errors in 11 (selected) bits can be tolerated.

∎

A7.8.2. Table of Type-B1 Codes

TABLE A7.8.1, below, gives the 'best' type-B1 code, for various selected values of J, together with the corresponding values of k, n_E, n_A & n_A/n_E. The actual constraint-length of the 'best' type-B1 codes is compared with that of rate-1/2 CSOCs constructed by Massey [18], or Wu [45]. The sixth column (marked "%") shows how much longer the type-B1 codes are, compared with the Massey or Wu ones.

TABLE A7.8.1

J	k	n_E	n_A	n_A/n_E	%
2 *	2	4	8	2.00	100.0
3	4	7	24	3.43	-
4 *	4	11	32	2.91	100.0
6 *	6	22	72	3.27	71.4
8	10	37	160	4.32	77.8
10 *	10	56	200	3.57	23.5
12 *	12	79	288	3.65	21.0
14	16	106	448	4.23	43.6
16 *	16	137	512	3.74	-
17	18	154	612	3.97	19.5
18 *	18	172	648	3.77	8.0
20	22	211	880	4.17	18.9
22 *	22	254	968	3.81	-
24	28	301	1,344	4.47	22.6
26	28	352	1,456	4.14	15.2
28 *	28	407	1,568	3.85	7.4
30 *	30	466	1,800	3.86	7.0
32	36	529	2,304	4.36	17.4
33	36	562	2,376	4.23	16.0
36 *	36	667	2,592	3.89	-
38	40	742	3,040	4.10	10.9
40 *	40	821	3,200	3.90	-
42 *	42	904	3,528	3.90	3.3
44	46	991	4,048	4.08	7.3
46 *	46	1,082	4,232	3.91	-
48	52	1,177	4,992	4.24	13.0
50	52	1,276	5,200	4.08	6.7
52 *	52	1,379	5,408	3.92	-
54	58	1,486	6,264	4.22	11.5
58 *	58	1,712	6,728	3.93	-
60 *	60	1,831	7,200	3.93	3.4
62	66	1,954	8,184	4.19	8.8
65	66	2,146	8,580	4.00	4.7
66 *	66	2,212	8,712	3.94	-
68	70	2,347	9,520	4.06	4.6
70 *	70	2,486	9,800	3.94	-

* Type-B1 codes meeting the lower bound on n_A [eqn (7.17)].

TABLE A7.8.1 (continued)

J	k	n_E	n_A	n_A/n_E	%
72 *	72	2,629	10,368	3.94	2.8
74	78	2,776	11,544	4.16	8.3
78 *	78	3,082	12,168	3.95	-
80	82	3,241	13,120	4.05	5.1
82 *	82	3,404	13,448	3.95	2.5
88 *	88	3,917	15,488	3.95	-
89	96	4,006	17,088	4.27	-
90	96	4,096	17,280	4.22	8.5
91	96	4,187	17,472	4.17	-
92	96	4,279	17,664	4.13	-
93	96	4,372	17,856	4.08	-
94	96	4,466	18,048	4.04	-
95	96	4,561	18,240	4.00	-
96 *	96	4,657	18,432	3.96	-
97	100	4,754	19,400	4.08	-
98	100	4,852	19,600	4.04	3.8
99	100	4,951	19,800	4.00	-
100 *	100	5,051	20,000	3.96	-
102 *	102	5,254	20,808	3.96	-
150 *	150	11,326	45,000	3.97	-
210 *	210	22,156	88,200	3.98	-
310 *	310	48,206	192,200	3.99	-
520 *	520	135,461	540,800	3.99	-
820 *	820	336,611	1,344,800	4.00	-
1,008 *	1,008	508,537	2,032,128	4.00	-
5,002 *	5,002	12,512,504	50,040,008	4.00	-
9,000 *	9,000	40,504,501	162,000,000	4.00	-
.....	.....	....	....	4 (?)	-

APPENDIX 7.9: OTHER CLASSES OF TYPE-B SELF-ORTHOGONAL CODES

A7.9.1. Proof of Theorem 7.13

Consider the initial array of a (2k,k,m) type-B self-orthogonal code. It is obvious that if any of the IA rows

* Type-B1 codes meeting the lower bound on n_A [eqn (7.17)].

are deleted (at random) the corresponding code will still be self-orthogonal (see Theorem 7.7). The minimum number of syndromes checking on any error bit, J (which equals m+1), will be reduced though and also (a compensation) the value of n-k will be reduced. Hence, if y ($1 \le y < k$) rows are deleted the IA generates a (2k-y,k,m) self-orthogonal type-B code. J', the new value of J, is unknown but it cannot be greater than J(1-y/k). This is so, because there will be J' copies of each of the k integers, hence at least kJ' integers in the IA, which has dimensions (n-k) x (m+1) = (k-y) x J, i.e. $(k-y)J \ge kJ' \Longrightarrow J' \le J(k-y)/k$.

QED

A7.9.2. Proof of Theorem 7.14

According to Definition 7.2 and the hypothesis:

$$a_{x,z} \equiv za_{x,1} = zx \pmod{k+1} \qquad (A)$$

From (A), $a_{x,z} + a_{k+1-x,z} \equiv zx + z(k+1-x) = z(k+1) \pmod{k+1}$

$\Longrightarrow \quad a_{x,z} + a_{k+1-x,z} \equiv 0 \pmod{k+1}$

$\Longrightarrow \quad k+1 \mid a_{x,z} + a_{k+1-x,z}$

$$\Longrightarrow \qquad a_{x,z} + a_{k+1-x,z} = q(k+1) \qquad (B)$$

Since $0 < a_{x,z}\ ,\ a_{k+1-x,z} < k+1 \Longrightarrow$

$0 < a_{x,z} + a_{k+1-x,z} < 2(k+1)$ and using (B), $0<q(k+1)<2(k+1)$

$\Longrightarrow 0<q<2 \Longrightarrow q=1$, and from (B): $a_{x,z} + a_{k+1-x,z} = k+1$.

From (A) & (B), $a_{x,z}-zx = q(k+1) \Longleftrightarrow a_{x,z} = zx+q(k+1)$. Let $p|x$. Then, since p is a divisor of k+1, p also divides $a_{x,z}$. Let p divide $a_{x,z}$. For the same reason, p divides zx. Now, z is a column number and as such $1 \le z \le J < p$ (see Theorem 7.10). Then, since $p>z$, p does not divide z, hence $(p,z)=1$ (according to Theorem A7.1.9), hence p divides x (according to Theorem A7.1.10).

Let b,z,w be as in the hypothesis and <u>assume</u> that there exist i & j, with $1 \le i \ne j < (k+1)/p$, such that $a_{b+ip,z}=a_{b+jp,w}$. Then, from (A):

$z(b+ip) \equiv a_{b+ip,z} = a_{b+jp,w} \equiv w(b+jp) \pmod{k+1} \quad \Longrightarrow$

$zb-wb \equiv wjp-zip \pmod{k+1} \Longrightarrow (z-w)b+(zi-wj)p = s(k+1)$

and since p divides k+1, it also divides (z-w)b. Since b<p, p does not divide b, hence (p,b)=1 (by Theorem A7.1.9), hence p divides z-w (by Theorem A7.1.10). Then, z-w = qp. But since z & w are IA columns, $0<z,w\le J<p \Longrightarrow -p<qp<p \Longrightarrow -1<q<1 \Longrightarrow q=0 \Longrightarrow z=w$. Hence, $a_{b+ip,z} = a_{b+jp,z} \Longleftrightarrow a_{b+ip,1} \equiv a_{b+jp,1} \pmod{(k+1)/(k+1,z)}$ (by Theorem 7.6). Since $z\le J<p$ (see Theorem 7.10) then, $(z,k+1)=1$ (by Theorem A7.1.8). Hence, $a_{b+ip,1} \equiv a_{b+jp,1} \pmod{k+1} \Longrightarrow a_{b+ip,1} = a_{b+jp,1}$ (because the two elements are in [1,k]). This <u>contradicts</u> Theorem 7.8 (iii), hence the 3rd result of the theorem.*

QED

A7.9.3. <u>Proof of Theorem 7.16</u>

Consider an (n,k,p-2) type-B self-orthogonal code. Each row contains p-1 distinct elements, while there are (k+1)/p-1 distinct multiples of p. Hence, the number of rows that are multiples of p (and will contain only likewise elements), say, x must be such that the number of elements in them, x(p-1), is at least J times the number of the distinct multiples of p: $x(p-1) \ge [(k+1)/p-1]J$. On the other hand, x cannot exceed the number of multiples of p, i.e. (k+1)/p-1. Hence, if $A \triangleq (k+1)/p-1$: $AJ/(p-1) \le x \le A$.

x is the number of rows that contain multiples of p only, while, according to Theorem 7.15, the rows that do not contain multiples of p are exactly J(k+1)/p. Their sum is the total number of rows of the resulting IA, which equals n-k. Then, bounds for n may be obtained:

$$AJ/(p-1)+J(k+1)/p \le x+J(k+1)/p = n-k \le A+J(k+1)/p$$

$$\Longrightarrow \quad AJ/(p-1)+J(A+1) \le n-k \le A+J(A+1)$$

$$\Longrightarrow \quad AJp/(p-1)+J+k \le n \le A(J+1)+J+k$$

$$\Longrightarrow \quad J[Ap/(p-1)+1]+k \le n \le [(k+1)/p-1](J+1)+J+k$$

$$\Longrightarrow \quad J[p(A+1)-1]/(p-1) \le n \le (J+1)(k+1)/p-J-1+J+k$$

$$\Longrightarrow \quad J[(k+1)-1]/(p-1) \le n \le (J+1)(k+1)/p-1+k$$

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

$\Longrightarrow$ $Jk/(p-1)+k \le n \le (J+1)(k+1)/p-1+k$

QED

A7.9.4. Proof of Theorem 7.18

Let the number of multiples of p [=(k+1)/p-1] be greater than the width of the array but not more than twice that width. In such a case, two rows are enough, if together they contain a distinct set of integers. Since the width of the row is p-1 and the number of multiples of p is (k+1)/p-1, then the condition on p & k is

$$p-1 < (k+1)/p-1 \le 2(p-1) \iff p < (k+1)/p \le 2p-1 \iff$$

$$p^2 < k+1 \le p(2p-1) \iff p < (k+1)/p \le 2p-1 \qquad (A)$$

Of course, k+1 must be an odd positive integer whose smallest prime factor is p. For example, if p=5, then $25 < k+1 \le 45$, hence the only possible value of k+1 is 35 (27, 33, 39 & 45 are divided by 3 and 29, 31, 37, 41 & 43 are primes).

It will be shown that a given value of p is suitable, only if (k+1)/p is a prime number. If (k+1)/p is a prime number, because $p < (k+1)/p$ [see (A), above] p is the smallest prime of k+1 and (A) is satisfied. If (k+1)/p is not a prime then it will have at least two prime factors, say q & r. <u>Assume</u> that both are not less than p. Then, $p^2 \le qr \le (k+1)/p \le 2p-1 < 2p \Longrightarrow p<2 \Longrightarrow$ <u>contradiction</u>, hence if (k+1)/p is not a prime it will have a prime factor less than p. Then, p & k should be such that (k+1)/p is also a prime, $(k+1)/p \triangleq q$. Equivalently, it is required that k+1 = pq, where q is a prime greater than p and less than 2p. For p=7, q should be >7 and <14, hence possible values for q are 11 & 13, giving a k equal to 76 & 90 respectively.

The number of rows with elements that are not multiples of p is J(k+1)/p (see Theorem 7.15). The number of rows with elements that are multiples of p, is 2J. Hence, n-k = $2J+J(k+1)/p = J(2p+k+1)/p \Longrightarrow n = k+J(2p+k+1)/p$.

The IA construction instructions will be similar to those for the type-B2 codes, except for the 1st-column elements that are multiples of p. For the type-B2 codes, each such row contained all the multiples of p, while for this class of codes, two such rows are required and the instruction set must specify the pairs. It will be shown that two rows, specifically one with 1st-column element pi [$1 \le i \le (k+1)/p-1 = q-1$] and another with 1st-column element (k+1)-pi, contain all the multiples of p once and $2p-(k+1)/p-1 = 2p-q-1$ of them, twice.

Consider 1st-column element x=pi /$1 \le i \le q-1$. Element k+1-x = pq-pi is also divisible by p, hence it also generates multiples of p. Let $a_{pi,1}$ = pi and $a_{k+1-pi,1}$ = k+1-pi. According to Theorem 7.14, $a_{pi,z} + a_{k+1-pi,z} = k+1 \Longleftrightarrow a_{pi,z} + a_{pq-pi,z} = pq$, for all z=1,2,...,p-1 (the width of the IA is p-1). It will be shown that the set

$$S_i \triangleq \{a_{pi,z}, a_{pq-pi,v} \ / z=1,2,..,p-1 \ \& \ v=1,2,..,q-p\} \qquad (B)$$

contains all the multiples of p, exactly once, for any value of i ($1 \le i \le q-1$).

Let i /$1 \le i < q$. Elements $a_{pi,z}$ /z=1,2,...,p-1 are all distinct multiples of p, since they constitute the row with 1st element pi. Similarly, elements $a_{pq-pi,v}$ /v=1,2,...,q-p are all distinct multiples of p because they constitute part of the row with 1st element pq-pi ($q \le 2p-1 \Longleftrightarrow q-p \le p-1$). It is reminded that the 1st-column elements are pi & pq-pi, respectively. The total number of elements in S_i is p-1+q-p = q-1 = (k+1)/p-1, i.e. as many as the multiples of p. If there are any duplications these will be between the two rows. <u>Assume</u> that there exist one z ($1 \le z \le p-1$) and one v ($1 \le v \le q-p$) such that

$$a_{pi,z} = a_{pq-pi,v} \Longleftrightarrow a_{pi,z} = pq - a_{pi,v} \quad \text{(by Theorem 7.14)}$$

$$\Longleftrightarrow a_{pi,z} \equiv za_{pi,1} = zpi = pq - a_{pi,v} \equiv pq - vpi \pmod{pq}$$

$$\Longleftrightarrow (z+v)pi \equiv 0 \pmod{pq}$$

$$\Longleftrightarrow \text{there exists integer } s\text{, such that } (z+v)pi = spq$$

$$\Longleftrightarrow (z+v)i = sq \Longleftrightarrow q \text{ divides } (z+v)i$$

Since the only divisors of q are 1 and q, (q,i)=1 because i is positive and less than q. Then, by Theorem A7.1.10, q divides z+v $\Longrightarrow$ $z+v \ge q$. But, from (B), $1 \le z \le p-1$ & $1 \le v \le q-p$, hence $2 \le z+v \le p-1+q-p=q-1 \Longrightarrow z+v < q \Longrightarrow$ <u>contradiction</u>, hence all the elements of S_i are distinct and the set contains all the multiples of p.

Since the elements of the row with 1st element pi together with the first q-p elements of the row with 1st element pq-pi are distinct, the remaining of the elements of the latter row will have duplicates (obviously in the first row, because each row contains a distinct set of elements - for the same reason there are no triplications, etc).

To generate J copies of each multiple of p, a row with first element pi is selected together with the row with first element k+1-pi. To avoid overlap, $2pi < k+1 \Longleftrightarrow 2i < q \Longleftrightarrow i < q/2$. Since q=odd, $i=1,2,\dots,(q-1)/2$. Hence J cannot exceed $(q-1)/2$.

QED

A7.9.5. Proof of Theorem 7.19

From inequality (7.9), $m+1 < (k+1)/(k+1,a_{x,1})$ for all $x=1,2,\dots,n-k$. Alternatively, a number x, between 1 and k, may be chosen for the 1st column of the IA provided that $m+1 < (k+1)/(k+1,x)$. If $d_1 \le m+1 < d_2$, then $(k+1)/(k+1,x) > m+1 \ge d_1$. Since $(k+1)/(k+1,x)$ is a divisor of k+1, greater than d_1, it may only be greater or equal to the next divisor, i.e. d_2, hence $d_2 \le (k+1)/(k+1,x) \Longleftrightarrow (k+1,x) \le (k+1)/d_2$. Alternatively, if the latter is true, $(k+1)/(k+1,x) \ge d_2 > m+1 \Longrightarrow m \le (k+1)/(k+1,x)-2$. Hence, if $d_1 \le m+1 < d_2$ then Theorem 7.3 is equivalent to $(k+1,x) \le (k+1)/d_2$. Hence, x may be any integer between 1 & k, provided that it is not divided by any divisor of k+1 greater than $(k+1)/d_2$.

The 2nd result concerns the number of copies of any particular integer α ($1 \le \alpha \le k$), included in the IA. According to the definition of type-B codes (see Definition 7.2), the element in column z ($1 \le z \le m+1$) and row with first element x is congruent modulo k+1 to the product xz.

Then the number of copies of α equals the number of solutions of the congruence $xz \equiv \alpha \pmod{k+1}$, where x & z are restricted according to the above.

QED

A7.9.6. Proof of Theorem 7.20

Let i denote an IA element ($i \in [1,k]$) and $e \triangleq (i,k+1)$. Let z denote an IA column ($z \in [1,m+1]$) and $d \triangleq (z,k+1)$. Finally, let x denote a first-column element of the IA ($x \in [1,k]$) and $f \triangleq (x,k+1)$. According to Theorem 7.19, i, z & x are related via congruence:

$$zx \equiv i \pmod{k+1} \qquad \text{(A)}$$

where x must not be a multiple of any divisor of k+1 greater than $(k+1)/d_2$, or the same the greatest divisor of k+1 which also divides x should not exceed $(k+1)/d_2$, or the same $f \le (k+1)/d_2$.

According to Theorem A7.2.5, if congruence (A) is to be solved for x, it has exactly d solutions [$d \triangleq (z,k+1)$] in the range [1,k] (which is also the range of x), if d divides i, and none if $d \nmid i$. If $d \mid i$, of the d solutions only those which satisfy $f \triangleq (x,k+1) \le (k+1)/d_2$ are retained, hence the number of copies of i varies between 0 and d. This proves the general statement of the theorem.

The remaining of the proof is an elaboration on the last paragraph.

From Theorem A7.2.5, if $d \mid i$ (i.e. if i is a multiple of d), congruence (A) has exactly d solutions (in the range [1,k]), given by

$$x = \varsigma + j(k+1)/d \quad /j=0,1,\ldots,d-1 \qquad \text{(B)}$$

Hence, a column z may only contain elements i which are multiples of d. Also, there may be up to d copies of an individual multiple of d, i, along column z [i.e. solutions of (A)]. A solution is acceptable (i.e. the corresponding copy of i will be included in column z), if $f \le (k+1)/d_2$, i.e. if

$$f = \gcd\left(\varsigma + j(k+1)/d, k+1\right) \le (k+1)/d_2 \qquad \text{(C)}$$

According to Theorems A7.2.5 & A7.2.7, ç is given by

$$\varsigma \equiv (i/d)(z/d)^{\phi((k+1)/d)-1} \pmod{(k+1)/d} \qquad \text{(D)}$$

where ç is unique modulo (k+1)/d (i.e. there is only one solution of congruence (D) in the range [1,(k+1)/d]).

The requirement that $e/d \le (k+1)/d_2$ and the case for $d=1$, will complete the proof. Nevertheless, they require the proof of $\gcd(x,(k+1)/d) = \gcd(f,(k+1)/d) = e/d$, if $d|i$.

If $d|i$, from congruence (D) and Definition A7.2.1, there exists integer c such that $ç = (i/d)(z/d)^\mu + c(k+1)/d$, where $\mu \triangleq \Phi[(k+1)/d]-1$. Hence, from (B), $x = (i/d)(z/d)^\mu + c(k+1)/d + j(k+1)/d = (i/d)(z/d)^\mu + q(k+1)/d$. So, if $d|i$ there exist integers q & μ, such that

$$x = (i/d)(z/d)^\mu + q(k+1)/d \qquad \text{(E)}$$

$(z,k+1) \triangleq d \iff$ [by Theorem A7.1.5] $\left(z/d,(k+1)/d\right) = 1$

$\Longrightarrow$ [by Theorem A7.1.11] $\left((z/d)^\mu,(k+1)/d\right) = 1 \qquad$ (F)

Let $\left((i/d)(z/d)^\mu,(k+1)/d\right) \triangleq h \qquad$ (G)

Because $d \triangleq (z,k+1)$ divides k+1 and i (the latter by hypothesis),

$$e \triangleq \left(i,k+1\right) = \left((i/d)d,[(k+1)/d]d\right) = |d|\left(i/d,(k+1)/d\right)$$

[by (A7.1.1c)] and since d is a gcd, i.e. nonnegative,

$$\left(i/d,(k+1)/d\right) = e/d \qquad \text{(H)}$$

Since $(e/d) \mid (i/d) \Longrightarrow (e/d) \mid (i/d)(z/d)^\mu$ and since $(e/d) \mid (k+1)/d$ [from (H)] $\Longrightarrow$

$$(e/d) \mid h \qquad \text{(I)}$$

Let $(h,(z/d)^\mu) \triangleq b$. Since $b \mid h$ and $h \mid (k+1)/d$, $\Longrightarrow b \mid (k+1)/d$. Also, $b \mid (z/d)^\mu$. Then, $b \mid ((z/d)^\mu,(k+1)/d) = 1$ [see (F)]. Hence b=1 and

$$(h,(z/d)^\mu) = 1 \qquad \text{(J)}$$

From (G) & (J) and Theorem A7.1.10, $h \mid (i/d)$, and by (G) $h \mid (k+1)/d$. Then, by (H),

$$h \mid (e/d) \qquad \text{(K)}$$

By (I) & (K), $h = e/d$, and by (G):

$$\left((i/d)(z/d)^\mu,(k+1)/d\right) = e/d \qquad \text{(L)}$$

By (L) & Theorem A7.1.12,

$$\left((i/d)(z/d)^\mu,(k+1)/d\right) = \left((i/d)(z/d)^\mu+q(k+1)/d,(k+1)/d\right) = e/d$$

[using (E)]:

$$\left(x,(k+1)/d\right) = e/d \qquad (M)$$

$$\left(f,(k+1)/d\right) = \left((x,k+1),(k+1)/d\right) =$$

$$= \left(x,(k+1,(k+1)/d)\right) \quad \text{[by (A7.1.1b)]}$$

$$= \left(x,(k+1)/d\right) \quad \text{[since } (k+1)/d \mid k+1\text{]}.$$

Using (M):

$$\left(x,(k+1)/d\right) = \left(f,(k+1)/d\right) = e/d \qquad (N)$$

Since $(e/d) \mid f \Longrightarrow (e/d) \le f$ and since f must be $\le (k+1)/d_2$, it is necessary that $e/d \le (k+1)/d_2$. Apart from the case of d=1, the proof is complete.

If z is relatively prime to k+1 (d=1), for a given i there is always exactly one solution of (A) (since $1 \mid i$). This solution, x, is the 1st-column element of the row which contains element i. This single solution is acceptable, only if $f=(x,k+1)\le(k+1)/d_2$. From (N), for d=1, $(x,k+1) = f = e$, so the condition $f \le (k+1)/d_2$ is equivalent to $e \le (k+1)/d_2$.

QED

APPENDIX 7.10: (n,k,k-1) TYPE-B SELF-ORTHOGONAL CODES

A7.10.1. Proof of Theorem 7.21

According to Theorem 7.3, if a row, say x, is not to contain a zero it is necessary and sufficient for its length not to exceed $(k+1)/(k+1,a_{x,1})-1$. Note that this implies also that $a_{x,1} \ne 0$, because $(k+1)/(k+1,0)-1 = (k+1)/(k+1)-1 = 0$, hence there exists no row if $a_{x,1}=0$.

To prove the first part:

If m+1=k, then every row should have length k (its maximum possible length). From Theorem 7.3:

$$(k+1)/(k+1,a_{x,1})-1 = k \quad \text{for all } x=1,2,\dots,n-k \quad \Longleftrightarrow$$

$$(k+1)/(k+1,a_{x,1}) = k+1 \quad \text{for all } x=1,2,\dots,n-k \quad \Longleftrightarrow$$

$$(k+1,a_{x,1}) = 1 \quad \text{for all } x=1,2,\dots,n-k$$

Conversely, if $(k+1,a_{x,1}) = 1$ for all $x=1,2,\ldots,n-k$, then $(k+1)/(k+1,a_{x,1})-1 = k$ for all $x=1,2,\ldots,n-k$, hence every row has length k, and $m+1=k$.

Hence, a necessary and sufficient condition for $m=k-1$, is condition (7.24).

To prove the second part:

Let (7.25) hold true. Then, the elements of the first column are incongruent to each other modulo any non-trivial divisor of k+1.

From Theorem A7.1.7:

$d|(k+1) \iff$ there exists c: $d = (k+1)/(k+1,c)$ $/1 \le c \le k+1$

$d|(k+1)$ $/d>1$ $\iff$ $d = (k+1)/(k+1,c)$ $/1 \le c \le k+1$ & $d>1$ (A)

$d = 1 \iff k+1 = (k+1,c) \iff k+1 \mid c$ (B)

From (A), $k+1 \ge c$. Hence, if $k+1 \mid c$ ($\Longrightarrow k+1 \le c$), then $k+1 = c$. Conversely, if $k+1 = c \Longrightarrow k+1 \mid c$. Hence:

Given $1 \le c \le k+1$: $k+1 \mid c \iff k+1 = c$, and using (B):

Given $1 \le c \le k+1$: $d = 1 \iff k+1 = c$

Given $1 \le c \le k+1$: $d \ne 1 \iff k+1 \ne c$

Given $1 \le c \le k+1$: $d \ne 1 \iff k+1 < c$ and since $d \ge 1$:

Given $1 \le c \le k+1$: $d > 1 \iff k+1 < c$. So, from (A):

$d|(k+1)$ $/d>1$ $\iff$ $d = (k+1)/(k+1,c)$ $/1 \le c < k+1$ (C)

Then, from (C) & (7.25), the elements of the 1st column of the IA are incongruent modulo $((k+1)/(k+1,c))$ for all c less than $k+1 = m+2$, hence for all $c \le m+1$. Then, by (7.13b), the code is self-orthogonal.

Consider the converse now. Let the code be self-orthogonal. Then, by the corollary of Theorem 7.5,

$$a_{r,1} \not\equiv a_{v,1} \left(\text{mod } (k+1)/(k+1,c)\right) \qquad \text{(D)}$$

for all r & v and for all elements $a_{r,u} = a_{v,w}$ of these two rows that are equal (where $1<u,w \le m+1$) and for all positive integers c less than u & w. Since $m+1=k$ and the rows are

made of distinct elements (see Theorem 7.6), any two rows, say x & y /$x \neq y$ & $1 \le x,y \le n-k$, contain the same set of elements (in different order). Let element i $1 \le i \le k$, be in position w_i in row v and in position u_i in row r, and let σ_i be the minimum between u_i & w_i. Then, c ranges through the integers $1,2,\ldots,\mathrm{MAX}\{\sigma_1,\sigma_2,\ldots,\sigma_k\}-1$.

<u>Assume</u> that there exist at least two rows, say r & v, such that their first elements ($a_{r,1}$ & $a_{v,1}$) are congruent modulo at least one divisor d, of k+1. Then:

$a_{r,1} \equiv a_{v,1} \pmod d \quad \Longrightarrow$

there exists integer q such that, $a_{r,1}-a_{v,1} = qd \quad \Longrightarrow$

$a_{r,1}\mu(k+1)/d - a_{v,1}\mu(k+1)/d = qd\mu(k+1)/d$, where $(\mu,k+1)=1$.

Then: $a_{r,1}\mu(k+1)/d \equiv a_{v,1}\mu(k+1)/d \pmod{k+1} \quad \Longrightarrow$

$$a_{r,z} \equiv a_{v,z} \pmod{k+1} \quad /z=\mu(k+1)/d,\ \mu=1,2,\ldots,d-1 \qquad \text{(E)}$$

Since k=even, k+1 is odd and its smallest prime factor, p, is ≥ 3. Then, $d \ge p \ge 3 \Longrightarrow d-1 \ge 2$. Then (E) is valid for at least $\mu=1,2$. From (E), for $\mu=2$, [note that $(2,k+1)=1$], $a_{r,s} \equiv a_{v,s} \pmod{k+1}$ /$s \triangleq 2(k+1)/d$. Since, by hypothesis, the code is self-orthogonal, from the corollary of Theorem 7.5 and for all $c=1,2,\ldots,2(k+1)/d-1$, $a_{r,1} \not\equiv a_{v,1} \pmod{(k+1)/(k+1,c)}$. If $c=(k+1)/d$, which is $\le 2(k+1)/d-1$ for all $k \ge 2$, then $(k+1)/(k+1,c) = d$, hence: $a_{r,1} \not\equiv a_{v,1} \pmod d$, which <u>contradicts</u> the assumption. So, there are no two rows whose first elements are congruent some divisor of k+1.

This proves the second part.

QED

A7.10.2. <u>Proof of Theorem 7.22</u>

Since m=k-1, each row must have length m+1=k. From the proof of Theorem 7.21, a necessary and sufficient condition is that all first-column elements are relatively prime to k+1 (note that this was proved without assuming that k is even).

$$\left(k+1,a_{x,1}\right) = 1 \qquad \text{(A)}$$

Since k+1 is even, from (A), $a_{x,1}$ must be odd. Then, the first-column elements must all be congruent to 1 (mod 2), so for any two rows with first elements, say, $a_{v,1}$ & $a_{r,1}$:

$a_{v,1} \equiv a_{r,1} \pmod 2 \iff 2 \mid (a_{v,1}-a_{r,1}) \iff$

$2(k+1)/2 \mid (a_{v,1}-a_{r,1})(k+1)/2 \iff$

$[(k+1)/2]a_{v,1} \equiv [(k+1)/2]a_{r,1} \pmod{k+1} \iff$

$$a_{v,z} \equiv a_{r,z} \pmod{k+1} \quad /z \hat{=} (k+1)/2 \qquad \text{(B)}$$

Since $a_{x,1} = 1$ is an acceptable first-column element, it follows that $a_{x,z} \equiv (k+1)/2 \pmod{k+1}$ and because $0<a_{x,z}<k+1$ $0\le|a_{x,z}-(k+1)/2|<k+1$, it follows from Theorem A7.2.3 that $a_{x,z} = z$ [$\hat{=}$ $(k+1)/2$]. Hence, from (B):

$$\text{For all } v=1,2,\ldots,n-k: \quad a_{v,z} = z \quad /z \hat{=} (k+1)/2 \qquad \text{(C)}$$

(C) proves the 3rd part of the theorem.

Next, it will be shown that if k+1=even, the IA may only have 2 rows, if the code is to be self-orthogonal.

Let any two rows, with first elements, say, $a_{v,1}$ & $a_{r,1}$. Since, by Theorem 7.6, the rows contain a distinct set of elements, because their range is [1,k] and since there are k of them, each row contains the integers 1,2,...,k, (in a unique order, of course). According to the corollary of Theorem 7.5, for the code to be self-orthogonal, the first elements of any two rows, say $a_{v,1}$ & $a_{r,1}$, must be incongruent modulo $(k+1)/(k+1,c)$ for all $c = 1,2,\ldots,\text{MAX}\{\sigma_1,\sigma_2,\ldots,\sigma_k\}-1$, where $\sigma_i \hat{=} \text{MIN}\{u_i,w_i\}$ and u_i & w_i are the positions of element i ($1\le i\le k$) in rows v & r, respectively.

The smallest divisor of k+1 is 2, hence the largest one is (k+1)/2. If (k+1)/2 is included in the range of values of c, then $(k+1)/(k+1,(k+1)/2) = 2$ and $a_{v,1}$ & $a_{r,1}$ must be incongruent modulo 2, which means that one of the two must be 0 (mod 2), i.e. an even integer. This is not permitted by (A) above, hence

A necessary condition for the code to be self-orthogonal is that (k+1)/2 is not included in the range of c. (D)

Consider rows with first elements $a_{v,1}$ & $a_{r,1}$. By (C), the middle element is (k+1)/2. Let A(v) denote the set of the (k-1)/2 elements in the first half of the row with 1st element $a_{v,1}$ and B(v) the (k-1)/2 elements in the second half of that row. Note that A(v) & B(v) together contain k-1 distinct elements [the kth is in column z $\hat{=}$ (k+1)/2].

For each element i ($1 \le i \le k$), either i=(k+1)/2 in which case it is in column (k+1)/2, or $i \in A(v)$, or $i \in B(v)$.

If i=(k+1)/2, then i is in column (k+1)/2 on each row v, r, etc. For this case, σ_i=(k+1)/2. Hence, condition (D) is not violated.

All the elements of B(v) must also belong to A(r) because if, say, j appears in B(r) then σ_j>(k+1)/2, hence c is allowed to range up to at least (k+1)/2, violating thus the necessary condition for self-orthogonality [see (D)]. Hence A(r)=B(v) and, by necessity, A(v)=B(r). So:

A necessary condition for the existence of a self-orthogonal type-B code with k+1=even, is that for any two rows with first-column elements $a_{v,1}$ & $a_{r,1}$: A(v)=B(r) and A(r)=B(v). (E)

Assume that the IA has at least three rows with first-column elements, say, $a_{v,1}$, $a_{r,1}$ & $a_{s,1}$. According to condition (E), above, for the code to be self-orthogonal it is necessary to have A(v)=B(r) & A(r)=B(v), and A(v)=B(s) & A(s)=B(v), and A(r)=B(s) & A(s)=B(r). From these six equations it follows that A(v)=B(r) & B(r)=A(s) & A(s)=B(v), which gives A(v)=B(v), which means that row v contains duplicate elements, which is a <u>contradiction</u>, by Theorem 7.6. Hence, the IA must have less than three rows if the code is to be self-orthogonal, i.e. $n-k \le 2$.

From the discussion so far, on type-B codes, it is apparent that since each row contains all integers in the range [1,k], each error bit $e_0^{(i)}$ appears in the syndrome equations exactly n-k times. Then, there are exactly n-k syndromes checking on each error bit, hence J=n-k and, since $n-k \le 2$, it follows that $J \le 2$. If the code is to have a non-zero guaranteed error-correcting capability (t>0), then J>1 (note, from Theorem 5.3, that t = $\lfloor J/2 \rfloor$). Hence, the only possible value for J is 2, and so is for n-k.

This <u>proves part of the fourth statement of the theorem (J=2)</u>.

. . . .

Next, it will be proved that a necessary condition [direct consequence of (E)] for the existence of a self-orthogonal code is that $a_{1,1}+a_{2,1}=k+1$. To achieve this it is necessary to show that, for all $v=1,2,...,n-k$ & all $b,c=1,2,..,k$,

$$ba_{v,c} \equiv a_{v,y} \text{ (mod } k+1\text{), where } y \equiv bc \text{ (mod } k+1\text{) \& } 1 \le y \le k \qquad \text{(F)}$$

For any $b,c,=1,2,...,k$ and any $v=1,2,...,n-k$, let $bc \equiv y$ (mod k+1), where $1 \le y \le k$ [$y = bc - \lfloor bc/(k+1) \rfloor (k+1)$]. Then, $(k+1) \mid (bc-y) \Longrightarrow (k+1) \mid (bc-y)a_{v,1}$. Hence:

For all $v=1,2,...,n-k$ & all $b,c=1,2,...,k$:

$$bca_{v,1} - ya_{v,1} \equiv 0 \text{ (mod } k+1\text{)} \Longrightarrow$$

$$b(ca_{v,1}) \equiv ya_{v,1} \text{ (mod } k+1\text{)}$$

Using Definition 7.2 and the fact that $1 \le c,y \le k$, (F) follows immediately.

Let $c = k+1-\mu$ in (F). Then, $\mu = k+1-c$ and for $c=1,2,..,k$ $\Longrightarrow \mu = k,k-1,..,1$. Since $y \equiv b(k+1-\mu) \equiv -b\mu$ (mod k+1):

For all $v=1,2,...,n-k$ & all $b,\mu=1,2,...,k$:

$$ba_{v,k+1-\mu} \equiv a_{v,y} \text{ (mod } k+1\text{), where } y \equiv -b\mu \text{ (mod } k+1\text{) \& } 1 \le y \le k \qquad \text{(G)}$$

Let $a_{1,1}$ & $a_{2,1}$ be the two elements of the 1st column. The elements of the 2nd half of the first row may be expressed by $a_{1,k+1-u}$ /$u=1,2,...,(k-1)/2$. Because B(1)=A(2) [by (E)], $a_{2,1}$ must equal one of $a_{1,k+1-u}$ /$u=1,2,...,(n-k)/2$. Let $a_{2,1} = a_{1,k+1-w}$ /$1 \le w \le (k-1)/2$. Then, the elements of A(2) are given by (see Definition 7.2):

$$a_{2,r} \equiv ra_{1,k+1-w} \text{ (mod } k+1\text{)} \quad \text{[and using (G)]} \Longrightarrow$$

$$a_{2,r} \equiv a_{1,v} \text{ (mod } k+1\text{), where } v \equiv -rw \text{ (mod } k+1\text{)} \qquad \text{(H)}$$

Since the IA elements are in the range (0,k+1), their difference $|a_{2,r}-a_{1,v}|$ is in the range [0,k). Then, by Theorem A7.2.3:

$$a_{2,r} = a_{1,v} \quad /v \equiv -rw \text{ (mod } k+1\text{)} \qquad \text{(I)}$$

For those values of r, for which $1 \le k+1-rw \le k$, and because $k+1-rw \equiv -rw$ (mod k+1), it follows that $v = k+1-rw$. Then, from (I):

$$a_{2,r} = a_{1,k+1-rw} \quad /1 \le k+1-rw \le k \qquad (J)$$

Notice from (J) that if w=1, $a_{2,1} = a_{1,k}$, $a_{2,2} = a_{1,k-1}, \ldots,$ $a_{2,z-1} = a_{1,z+1}$, where z $\hat{=}$ (k+1)/2. Hence, w=1 is a suitable value. It will be shown that any other value of w (with w∈[1,(k-1)/2]) will result in a violation of condition (E).

Let w>1. As r increases, k+1-rw decreases. By design, element $a_{2,1} = a_{1,k+1-w}$ is in the 2nd half of the first row. It is not known though if $a_{2,2} = a_{1,k+1-2w}$ appears in the 2nd half of the 1st row, or not. The same applies to $a_{2,3}$, etc.

Note that elements $a_{2,1}, a_{2,2}, \ldots, a_{2,x}$, where x $\hat{=}$ (k-1)/2, equal elements $a_{1,k+1-w}, a_{1,k+1-2w}, \ldots, a_{1,k+1-xw}$, where if k+1-jw becomes negative an adequate number of (k+1)s is added so that it becomes positive and not greater than k. Hence, the 1st half of the 2nd row is identical to a reversed & 'interleaved' (with 'degree' w) first row.

Consider, for <u>example</u>, k=19 and eight rows (only two to be retained), with first elements 1, 3, 7, 9, 11, 13, 17 & 19 (all relatively prime to k+1=20).

1st half										*2nd half*								
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
3	6	9	12	15	18	1	4	7	10	13	16	19	2	5	8	11	14	17
7	14	1	8	15	2	9	16	3	10	17	4	11	18	5	12	19	6	13
9	18	7	16	5	14	3	12	1	10	19	8	17	6	15	4	13	2	11
11	2	13	4	15	6	17	8	19	10	1	12	3	14	5	16	7	18	9
13	6	19	12	5	18	11	4	17	10	3	16	9	2	15	8	1	14	7
17	14	11	8	5	2	19	16	13	10	7	4	1	18	15	12	9	6	3
19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Select any of the eight rows as the IA's first row, say the 3rd row* (starting with element $a_{1,1}$=7). Candidates for the other row may be found in the 2nd half of this row: 13(w=1), 6(w=2), 19(w=3), 12(w=4), 5(w=5), 18(w=6), 11(w=7), 4(w=8) and 17(w=9). Note though that from these candidates one must exclude all elements not relatively prime to k+1=20. Hence the acceptable list of rows** is 13(w=1), 19(w=3), 11(w=7) and 17(w=9).

Let $a_{2,1}$ = 19 (w=3). Then the first half of the IA's 2nd row will be 19 18 17 16 15 14 13 12 11. According to the

* Highlighted heavily.
** Highlighted.

theory, these 9 elements should equal elements $a_{1,20-3r}$, where $r=1,2,...,9$ and where $20-3r$ is kept within $[1,19]$ by adding 20s whenever necessary. The 1st half of the 2nd row, corresponds to the 1st-row elements $a_{1,17}$, $a_{1,14}$, $a_{1,11}$, $a_{1,8}$, $a_{1,5}$, $a_{1,2}$, $a_{1,19}$ $(=a_{1,-1})$, etc, $a_{1,13}$ $(=a_{1,-7})$. To illustrate this, the two rows are arranged again below, with the first six elements (of the 2nd row) highlighted:

7	**14**	1	8	**15**	2	9	**16**	3	10	**17**	4	11	**18**	5	12	**19**	6	13
19	**18**	**17**	**16**	**15**	**14**	13	12	11	10	9	8	7	6	5	4	3	2	1

For condition (E) to be satisfied (a necessary condition, if the code is to be self-orthogonal), no highlighting should appear in the 1st half of the 1st row (the case above does not qualify). To put it otherwise, w should be chosen so that (E) is satisfied; in fact, all such values of w must be obtained, in order to arrive at all possible self-orthogonal codes.

All first-column elements must be relatively prime to $k+1$: $(a_{1,1},k+1) = (a_{2,1},k+1) = 1$, and since $a_{2,1} = a_{1,k+1-w}$, then $(a_{1,k+1-w},k+1) = 1 \Longrightarrow (a_{1,1}(k+1-w)-q(k+1),k+1) = 1 \Longrightarrow (-wa_{1,1}+s(k+1),k+1) = 1 \Longrightarrow (-wa_{1,1},k+1) = 1$ (by Theorem A7.1.12). Then, $(w,k+1)$ may only be 1, because otherwise $(-wa_{1,1},k+1)$ would not be 1 either. Hence,

$$(w,k+1) = 1 \qquad \text{(K)}$$

The 1st element $(a_{1,k+1-w})$ is somewhere in the 2nd half of row 1, the 2nd element $(a_{1,k+1-2w})$ maybe in the 2nd half, or it may not be, and so on, but it seems that at least one of the elements will be in the 1st half of the 1st row (forbidden region). Let $a_{1,k+1-1w}, a_{1,k+1-2w}, ..., a_{1,k+1-xw}$ be the first x elements of the 1st half of the 2nd row that are all in the 2nd half of the 1st row, as well. Then, $k+1-xw > (k+1)/2 \Longleftrightarrow (k+1)/2 > xw \Longleftrightarrow x < (k+1)/(2w)$. From (K), w $(w>1)$ does not divide $(k+1)/2$ because if it did, $k+1 = 2qw$ and then $(w,k+1) = (w,2qw) = w > 1$. Then, the maximum value of x is $\lfloor(k+1)/(2w)\rfloor$. Then, element $a_{1,k+1-Xw}$ $(X \hat{=} x_{max})$ is in the 2nd half, but element $a_{1,k+1-(X+1)w}$ will be in the 1st half, provided that $k+1-(X+1)w > 0$ $/X+1 = x_{max}+1 = \lfloor(k+1)/(2w)\rfloor+1$.

But, $k+1-(X+1)w = (k+1)-\lfloor(k+1)/(2w)\rfloor w-w = (k+1)/2 -$

$\lfloor(k+1)/(2w)\rfloor w + (k+1)/2-w = MOD[(k+1)/2,w] + [(k+1)/2-w]$, where $MOD(A,B) \triangleq A$ modulo B, with $0 \le MOD(A,B) \le B-1$. Then, $k+1-(X+1)w \ge (k+1)/2-w \ge (k+1)/2-(k-1)/2 = 1$. Hence:

If $w>1$, there exists an element $a_{2,v}$ from the 1st half of the 2nd row $[v<(k+1)/2]$ which appears in the 1st half of the 1st row:

$$a_{2,v} = a_{1,k+1-vw} \quad /v \triangleq \lfloor(k+1)/(2w)\rfloor+1 \qquad (L)$$

Hence, if the code is to be self-orthogonal, w may only be equal to 1. Then the 1st element of the 2nd row equals a_{k+1-1}, i.e. the first element of the 2nd row must be equal to the last element of the first row:

$$a_{2,1} = a_{1,k} \qquad (M)$$

Since $a_{1,k} \equiv ka_{1,1} \equiv (k+1)a_{1,1}-a_{1,1} \equiv -a_{1,1} \pmod{k+1}$

then, $a_{1,k} + a_{k,1} \equiv 0 \pmod{k+1}$. From (M):

$a_{1,1} + a_{2,1} \equiv 0 \pmod{k+1} \Longrightarrow$ "there exists integer q such that $a_{1,1} + a_{2,1} = q(k+1)$". Since $a_{1,1}$ & $a_{2,1}$ are IA elements, it follows that $0 < a_{1,1} + a_{2,1} < 2(k+1) \Longrightarrow 0 < q(k+1) < 2(k+1) \Longrightarrow q=1 \Longrightarrow$

A necessary condition for the code to be self-orthogonal:

$$a_{1,1} + a_{2,1} = k+1 \qquad (N)$$

Note that from (N), and because $(a_{1,1},k+1)=1$, it also follows that $(a_{2,1},k+1) = (k+1-a_{1,1},k+1) = (-a_{1,1},k+1)$ (from Theorem A7.1.12) and finally, $(a_{2,1},k+1) = 1$.

This proves the 2nd statement of the theorem.

So far it has been proved that the following conditions are necessary, if k+1=even and the code is to be self-orthogonal:

1. $J = n-k = 2$,
2. $(a_{1,1},k+1) = 1$ and
3. $a_{2,1} = k+1-a_{1,1}$

Consider now a (k+2,k,k-1) type-B code with k+1=even and its IA with $a_{1,1}$ such that $(a_{1,1},k+1) = 1$ and $a_{2,1} = k+1-a_{1,1}$.

Because $(a_{1,1},k+1) = 1 \Longrightarrow (k+1-a_{1,1},k+1) = 1$ and since both rows satisfy Theorem 7.3, then they contain no zeros.

Let $a_{1,1}=x$. Then, the first row elements are given by

$$a_{1,r} \equiv rx \pmod{k+1} \quad /r=1,2,\dots,k \qquad (O)$$

while the 2nd row elements are given by

$$a_{2,r} \equiv (k+1-x)r \equiv -rx \pmod{k+1} \qquad (P)$$

From (O) & (P),

$$a_{1,r} + a_{2,r} \equiv 0 \pmod{k+1} \quad /r=1,2,\dots,k \quad \Longrightarrow$$

$$a_{1,r} + a_{2,r} = k+1 \quad /r=1,2,\dots,k \qquad (A7.10.1)$$

because $a_{1,r}$ & $a_{2,r}$ are IA elements, hence they are confined in the range $(0,k+1)$, so their sum cannot be less than 1, or greater than 2k. Also, from (P) and with the same reasoning:

$$a_{1,k+1-r} \equiv (k+1-r)x \equiv -rx \equiv a_{2,r} \pmod{k+1} \quad /r=1,2,\dots,k$$

$$\Longrightarrow \quad a_{2,r} = a_{1,k+1-r} \qquad (A7.10.2)$$

From (A7.10.2), the pairs of equal elements in the two rows appear in positions 1 & k, 2 & k-1, ..., $(k+1)/2$ & $(k+1)/2$, ..., k & 1. Hence, the beginning of the rightmost pair of equal elements is $(k+1)/2$, hence $c=1,2,\dots,(k+1)/2-1$ and according to the corollary of Theorem 7.5 the code is self-orthogonal if

$$a_{1,1} \not\equiv a_{2,1} \left(\mathrm{mod}\ (k+1)/(k+1,c)\right) \quad /c=1,2,\dots,(k+1)/2-1$$

Since $c<(k+1)/2$, it follows that $(k+1,c) < (k+1)/2 \Longrightarrow (k+1)/(k+1,c) > 2$. Hence it is required that $a_{1,1} = x$ and $a_{2,1} = k+1-x$ are incongruent modulo any divisor d of k+1, greater than 2.

<u>Assume</u> that there exists a divisor d of k+1, greater than 2, for which x & k+1-x are congruent modulo d:

$x \equiv k+1-x \pmod d \Longrightarrow$ there exists q such that

$k+1-2x = qd \Longrightarrow$ d divides 2x, because $d \mid k+1$.

Let $(d,x) \triangleq f$. Then, $f \mid d \mid k+1$ and $f \mid x$, hence $f \mid (x,k+1) = 1 \Longrightarrow f=1$. So, d divides 2 (by Theorem A7.1.10) $\Longrightarrow d\le 2 \Longrightarrow$ <u>contradiction</u> $\Longrightarrow$ there does not exist a divisor of k+1, greater than 2, such that $a_{1,1}$ & $a_{2,1}$ to be congruent. Then the code is self-orthogonal.

So, if k+1=even, a necessary and sufficient condition for the (k+2,k,k-1) code to be self-orthogonal is J=n-k=2, $a_{2,1}$ = k+1-$a_{1,1}$ and ($a_{1,1}$,k+1) = 1. This proves the main body of the theorem and also statement 4.

Regarding statement 1, it has already been proved that $a_{1,1}$ is the IA's generating element, and that $a_{1,1}$ may be any positive integer, not exceeding k, and is relatively prime to k+1. Since $a_{1,1}$=k+1 would not be considered because it would not be relatively prime to k+1, there are exactly Φ(k+1) such $a_{1,1}$s (see Definition A7.1.4), hence as many IAs.

QED

A7.10.3. Proof of Theorem 7.23

Let k+1 be an odd positive integer, with p its smallest prime factor. If the elements of the first column of the IA form a subset of

$$B(\mu) \hat{=} \left\{b_j \;/j=1,2,\ldots,p-1,\; b_j \equiv \mu j \pmod{k+1},\; 1 \le b_j \le k\right\} \qquad (A)$$

where (μ,k+1) = 1, then for each $a_{x,1}$ there exists j /1≤j≤p-1 such that:

$a_{x,1} = b_j \equiv \mu j \pmod{k+1} \implies$

there exists q such that $a_{x,1} = \mu j + q(k+1) \implies$

$\left(a_{x,1}, k+1\right) = \left(\mu j+q(k+1), k+1\right) = \left(\mu j, k+1\right)$ (by Theorem A7.1.12)

Since (μ,k+1) = 1 & (j,k+1) = 1 (because 1≤j<p),

then (μj,k+1) = 1 (by Theorem A7.1.13).*

It follows then that each row of the IA may have k elements (see Theorem 7.21), which are distinct (see Theorem 7.6). Hence, the IA contains exactly n-k copies of each element i=1,2,...,k, which is equivalent to the existence of exactly n-k syndromes checking on each error bit $e_0^{(i)}$ /i=1,2, ...,k (see Theorem 7.1). Hence, J = n-k.

Furthermore, if k+1 ≤ n ≤ k+p-1, then 2 ≤ n-k ≤ p-1, hence the number of rows may not exceed p-1.

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

Let any two elements of $B(\mu)$, say b_i & b_j, and assume that there exists a non-trivial divisor of k+1, say, d such that:

$$b_i \equiv b_j \pmod d \quad \Longleftrightarrow \quad \mu i \equiv \mu j \pmod d \quad \text{[by (A)]}$$

$$\Longleftrightarrow \quad i \equiv j \left(\bmod\ d/(\mu,d)\right) \quad \text{(by Theorem A7.2.4)}$$

Let $(\mu,d) \triangleq f$. Then, $f \mid d \mid (k+1)$ and $f \mid \mu$, hence $f \mid (\mu,k+1) = 1 \Longrightarrow f = 1$. So, $i \equiv j \pmod d \Longrightarrow$ d divides $|i-j| \Longrightarrow d \le |i-j|$.

But $1 \le i,j \le p-1 \Longrightarrow 0 < |i-j| < p-1 < d$, hence contradiction, hence the elements of $B(\mu)$ are incongruent to each other modulo any non-trivial divisor of k+1. This proves the existence part of the theorem.

To prove that the (n,k) type-B5 code is the only (n,k,k-1) type-B self-orthogonal code with k=even, it is enough to start with an (n,k,k-1) type-B self-orthogonal code.

Since the length of each row is k, it follows from Theorem 7.21 that

$$\left(a_{x,1}, k+1\right) = 1 \quad /x=1,2,\dots,n-k \qquad \text{(B)}$$

Also, for the same reason, the IA contains exactly n-k copies of each integer i /i=1,2,...,k, hence J = n-k (see Theorem 7.1).

Since the code is self-orthogonal, by Theorem 7.21, the $a_{x,1}$s are incongruent to each other modulo any nontrivial divisor, d, of k+1. Hence, since there cannot be more than d incongruent numbers modulo d, for any non-trivial divisor of k+1, and since p is the smallest of ds, then there are not more than p incongruent numbers modulo any divisor of k+1. Of these p numbers, one has to be excluded because it is a multiple of p [$\equiv 0 \pmod p$] and it would violate (B). Then,

$$1 \le n-k \le p-1 \quad \Longrightarrow \quad k+1 \le n \le k+p-1$$

QED

APPENDIX 7.11: GENERAL PROPERTIES OF TYPE-C CODES

A7.11.1. Proof of Theorem 7.24

Consider a cyclically-decodable (n,k,m) type-B code and two error bits, say, $e_h^{(\alpha)}$ and $e_h^{(\beta)}$ ($\alpha \neq \beta$), that satisfy the requirement laid by Definition 7.4. Let

$$s_{h+z(j)-1}^{\{x(j)\}} \quad /j=1,2,\ldots,J_\alpha, \quad 1\le x(j)\le n-k \quad \& \quad 1\le z(j)\le m+1 \qquad (A)$$

be the syndromes checking on $e_h^{(\alpha)}$. Then (see Definition 7.4) the syndromes checking on $e_h^{(\beta)}$ must be:

$$s_{h+z(j)}^{\{x(j)\}} \quad /j=1,2,\ldots,J_\beta, \quad 1\le x(j)\le n-k \quad \& \quad 1\le z(j)\le m+1 \qquad (B)$$

where, without loss of generality, a leftward shift has been assumed.

Note at first that, a necessary condition is $J_\alpha = J_\beta \triangleq J$. Also, from Theorem 7.1, because [see (A)] $s_{h+z(j)-1}^{\{x(j)\}}$ checks on $e_h^{(\alpha)}$,

$$a_{x(1),z(1)} = a_{x(2),z(2)} = \cdots = a_{x(J),z(J)} = \alpha \qquad (C)$$

and because [see (B)] $s_{h+z(j)}^{\{x(j)\}}$ checks on $e_h^{(\beta)}$,

$$a_{x(1),z(1)+1} = a_{x(2),z(2)+1} = \cdots = a_{x(J),z(J)+1} = \beta \qquad (D)$$

From (C) & (7.8):

$$z(1)a_{x(1),1} \equiv z(2)a_{x(2),1} \equiv \cdots \equiv z(J)a_{x(J),1} \pmod{k+1} \qquad (E)$$

From (D) & (7.8):

$$z(1)a_{x(1),1}+a_{x(1),1} \equiv z(2)a_{x(2),1}+a_{x(2),1} \equiv \cdots$$
$$\equiv z(J)a_{x(J),1}+a_{x(J),1} \pmod{k+1} \qquad (F)$$

It follows from (E), (F) & Theorem A7.2.2 that a necessary condition for the existence of a horizontal-shift cyclically decodable type-B code is

$$a_{x(1),1} \equiv a_{x(2),1} \equiv \cdots \equiv a_{x(J),1} \pmod{k+1} \qquad (G)$$

Because $1\le a_{x(j),1}\le k$, then $0\le |a_{x(v),1}-a_{x(w),1}|<k+1$, hence, by Theorem A7.2.3:

$$a_{x(1),1} = a_{x(2),1} = \cdots = a_{x(J),1} \qquad (H)$$

Then, by the corollary of Theorem 7.5, the code is not self-orthogonal.

QED

A7.11.2. Proof of Relations (7.27)

Following the same approach, as above, consider a cyclically-decodable (n,k,m) type-B code and two error bits, say, $e_h^{(\alpha)}$ & $e_h^{(\beta)}$ ($\alpha \neq \beta$), that satisfy the requirement laid by Definition 7.4. Let

$$s_{h+z(j)-1}^{\{x(j)\}} \quad /j=1,2,\ldots,J_\alpha, \quad 1 \le x(j) \le n-k \quad \& \quad 1 \le z(j) \le m+1 \qquad (A)$$

be the syndromes checking on $e_h^{(\alpha)}$. Then (see Definition 7.4) the syndromes checking on $e_h^{(\beta)}$ must be:

$$s_{h+z(j)-1}^{\{x(j)\pm 1\}} \quad /j=1,2,\ldots,J_\beta, \quad 1 \le x(j) \le n-k \quad \& \quad 1 \le z(j) \le m+1 \qquad (B)$$

where an upward or downward shift has been assumed.
As before, it is necessary that $J_\alpha = J_\beta \hat{=} J$.

From Theorem 7.1, because [see (A)] $s_{h+z(j)-1}^{\{x(j)\}}$ checks on $e_h^{(\alpha)}$,

$$a_{x(1),z(1)} = a_{x(2),z(2)} = \cdots = a_{x(J),z(J)} = \alpha \qquad (C)$$

and because [see (B)] $s_{h+z(j)-1}^{\{x(j)\pm 1\}}$ checks on $e_h^{(\beta)}$,

$$a_{x(1)\pm 1,z(1)} = a_{x(2)\pm 1,z(2)} = \cdots = a_{x(J)\pm 1,z(J)} = \beta \qquad (D)$$

From (C) & (7.8):

$$z(1)a_{x(1),1} \equiv z(2)a_{x(2),1} \equiv \cdots \equiv z(J)a_{x(J),1} \pmod{k+1} \qquad (E)$$

From (D) & (7.8):

$$z(1)a_{x(1)\pm 1,1} \equiv z(2)a_{x(2)\pm 1,1} \equiv \cdots \equiv z(J)a_{x(J)\pm 1,1} \pmod{k+1} \qquad (F)$$

(E) & (F) are relns (7.27).

QED

A7.11.3. Proof of Theorem 7.25

Assume that there exists a row, say, x ($1 \le x \le n-k$) of the SYRE which contains at least two syndrome bits, say, $s_{h+\alpha-1}^{(x)}$ & $s_{h+\beta-1}^{(x)}$ checking on the same error bit, say, $e_h^{(i)}$. Then, by Theorem 7.1, $a_{x,\alpha} = a_{x,\beta} = i$, which contradicts Theorem 7.6. This proves part of the 1st statement of the theorem.

<u>Assume</u> that there exists a column of the SYRE, say, z ($1 \le z \le m+1$) which contains at least two syndrome bits, say, $s_{h+z-1}^{(\alpha)}$ & $s_{h+z-1}^{(\beta)}$ checking on the same error bit, say, $e_h^{(i)}$. Then, by Theorem 7.1, $a_{\alpha,z} = a_{\beta,z} = i$, which is acceptable by Theorem 7.6 but not by Definition 7.5, as will be shown below:

Let syndrome bits, $s_{h+z-1}^{(x)}$ & $s_{h+z-1}^{(x+\delta)}$, where $1 \le x < x+\delta \le n-k \triangleq S$ & $1 \le z \le m+1$, check on error bit $e_h^{\{c(1)\}}$ /$1 \le c(1) \le k$.

Since $0<x<x+\delta \le S \Longrightarrow 0<\delta \le S-x \le S-1 \Longrightarrow 0<\delta<S$ (A)

Then, according to Definition 7.5, the two syndrome bits immediately below (without loss of generality, one of the two directions - upwards or downwards - has been chosen) $s_{h+z-1}^{(x)}$ & $s_{h+z-1}^{(x+\delta)}$, in the SYRE, must check on another error bit. In general, the pairs of syndrome bits

$$s_{h+z-1}^{(x+j-1)} \ \& \ s_{h+z-1}^{(x+\delta+j-1)} \quad /j=1,2,\ldots,S \qquad \text{(B)}$$

where $x+j-1$ & $x+\delta+j-1$ are kept within [1,S], by reducing them modulo S,

check on error bits $e_h^{\{c(j)\}}$ /$j=1,2,\ldots,S$, respectively.

From the above & Theorem 7.1:

$$a_{x+j-1,z} = a_{x+\delta+j-1,z} = c(j) \quad /j=1,2,\ldots,S \qquad \text{(C)}$$

It will be shown now that the $c_1, c_2, \ldots, c_S$ are not distinct.

Let $i \in [1,S]$ and let $j=i-\delta$ if $i>\delta$ and $j=S+i-\delta$ if i is otherwise. In both cases, $j \in [1,S]$ because: If $i>\delta$, $0<\delta<i \le S \Longrightarrow 0<i-\delta \le S-\delta \le S-1$ [by (A)] $\Longrightarrow 1 \le i-\delta<S \Longrightarrow j \in [1,S)$. If $i \le \delta$, then: $i-\delta \le 0 \Longrightarrow S+i-\delta \le S$, while $i-\delta \ge 1-(S-1)$ [see (A)] $\Longrightarrow 2-S \le i-\delta \Longrightarrow 2 \le S+i-\delta$, hence $j \in [2,S]$. Furthermore, $j \equiv i-\delta \pmod S$, while $j \ne i$ [otherwise, $\delta=0$ or $\delta=S$, both of which contradict (A)]. So:

For each $i \in [1,S]$, there exists $j \in [1,S]$ /$j \ne i$:

$$j \equiv i-\delta \pmod S \qquad \text{(D)}$$

From (D): $j \equiv i-\delta \pmod S \Longleftrightarrow$

$$x+i-1 \equiv x+\delta+j-1 \pmod S \qquad \text{(E)}$$

From (C):

$$a_{x+j-1,z} = a_{x+\delta+j-1,z} = c(j) \qquad \text{(F)}$$

$$a_{x+i-1,z} = a_{x+\delta+i-1,z} = c(i) \qquad (G)$$

From (E), (F) & (G), for each $c(j)$ /$j=1,2,\ldots,S$, there exists at least one $i\in[1,S]$: $c(i) = c(j)$. Hence, a complete cyclic shift can only be used to decode less than $S \triangleq n-k$ error bits, which <u>contradicts</u> Definition 7.5. Hence, no column of the SYRE contains more than one syndromes checking on the same error bit. This <u>concludes the proof of the 1st part of the theorem</u>.

Consider column w ($1\leq w\leq m+1$) of the SYRE. From Fig. 7.1, this contains the syndrome bits $s^{(i)}_{m+1-w}$, $i=1,2,\ldots,n-k$ ($h=0$). According to Theorem 7.1, these check on error bit $e^{(\alpha)}_0$, iff $a_{i,m+2-w} = \alpha$. Since each column of the SYRE contains syndromes checking on a different error bit, no two of $a_{i,m+2-w}$ must be equal, for $i=1,2,\ldots,n-k$. Hence, by necessity, no IA column must contain duplicate elements.

Furthermore, there must be another, say, J_w-1 SYRE columns containing syndrome bits checking on exactly the same error bits as column w. Hence, there must be J_w IA columns containing exactly the same elements. All these J_w IA columns form a *coset*. The *coset leader* is the column with the smallest column number. The *first coset* is the one with the 1st column as leader. Since each column contains exactly n-k elements, then each coset contains n-k distinct elements. If there are x cosets, then all of them contain x(n-k) distinct elements. For the code to check on each error bit, x(n-k) = k, from which it follows that n-k must divide k and also that there are k/(n-k) cosets.

$$x = k/(n-k) \qquad (H)$$

If the syndrome bits of a column of SYRE check on a certain sequence of error bits then, by Definition 7.5, the syndrome bits of another column of SYRE, belonging to the same coset, must check on the same sequence of error bits, the sequence starting from a different row this time. This means that (by Theorem 7.1), each column of the same coset must be a cyclic shift of some other column of that coset.

Let J_i /$1\leq i\leq x$ be the number of columns in coset i. Then, since the IA has m+1 columns,

$$J_1 + J_2 + \cdots + J_x = m+1 \qquad \text{(I)}$$

Furthermore, by Theorem 7.1, the number of syndrome bits checking on error bit $e_0^{(i)}$ /$1 \le i \le k$ equals the number of is in the IA. But since, by Theorem 7.6, no IA row contains duplicate elements, there are no more than n-k is in the IA, hence there are no more than n-k syndromes checking on $e_0^{(i)}$.

So $$J_i \le n-k \quad \text{for all } i=1,2,\ldots,x \qquad \text{(J)}$$

This <u>proves the 2nd part of the theorem</u>.

Let $a_{1,1} \hat{=} \alpha$ and a column β /$1 \le \beta \le m+1$ belonging to the 1st coset. Then, $a_{1,\beta} \equiv \beta\alpha \pmod{k+1}$. Since $\beta\alpha$ belongs to the first coset it must also appear in the first column, as well, hence there must exist $a_{x,1} \equiv \beta\alpha \pmod{k+1}$. In general, if $S \hat{=} n-k$:

If $a_{1,1} \hat{=} \alpha \quad \Longrightarrow \quad a_{1,\beta} \equiv \beta\alpha \pmod{k+1}$

$\Longrightarrow$ there exists $x(2) \in [1,S]$: $a_{x(2),1} \equiv \beta\alpha \pmod{k+1}$

$\Longrightarrow \quad a_{x(2),\beta} \equiv \beta^2\alpha \pmod{k+1}$

$\Longrightarrow$ there exists $x(3) \in [1,S]$: $a_{x(3),1} \equiv \beta^2\alpha \pmod{k+1}$

$\Longrightarrow \quad a_{x(3),\beta} \equiv \beta^3\alpha \pmod{k+1}$

$\Longrightarrow$ there exists $x(4) \in [1,S]$: $a_{x(4),1} \equiv \beta^3\alpha \pmod{k+1}$

$\Longrightarrow \quad a_{x(4),\beta} \equiv \beta^4\alpha \pmod{k+1}$

. . . .

$\Longrightarrow$ there exists $x(S) \in [1,S]$: $a_{x(S),1} \equiv \beta^{S-1}\alpha \pmod{k+1}$

$\Longrightarrow \quad a_{x(S),\beta} \equiv \beta^S\alpha \pmod{k+1}$

So far, column 1 has the S elements $\alpha, \beta\alpha, \beta^2\alpha, \ldots, \beta^{S-1}\alpha$ (all reduced modulo k+1 in the range [1,k+1]). For the code to be self-orthogonal, they should be distinct (see Theorem 7.5).

Also, column β has the S elements $\beta\alpha, \beta^2\alpha, \ldots, \beta^{S-1}\alpha, \beta^S\alpha$ (all reduced modulo k+1 in the range [1,k+1]). For the code to be self-orthogonal, they should be distinct (see Theorem 7.5).

A direct consequence of the requirement for the elements of each of these two columns to be distinct is that

$$\beta^i \not\equiv 1 \pmod{k+1} \quad /i=1,2,\ldots,S-1 \qquad \text{(K)}$$

Furthermore, the two columns must have the same elements. Since they differ only in α & $\beta^S\alpha$, it is required that:

$$\beta^S\alpha \equiv \alpha \pmod{k+1} \qquad \text{(L)}$$

Finally, note that there must exist one such column β, of the 1st coset because otherwise J_1=1, and the J of the code will be one, hence the code will have zero error-correcting capability.

This proves the 3rd part of the theorem.

QED

A7.11.4. Proof of Lemma 7.3

Consider the 3rd statement of Theorem 7.25, and in particular the relation among α, β & k+1. Let $S \mathrel{\hat{=}} n-k$:

From (7.29b): $\alpha(\beta^S-1) \equiv 0$ (mod (k+1). Two obvious solutions are $\alpha \equiv 0$ & $\beta^S \equiv 1 \pmod{k+1}$. The first is not acceptable, while the 2nd is not always possible. Let us consider all solutions of the congruence. By Theorem A7.2.5:

If $(\beta^S-1,k+1) \mathrel{\hat{=}} s$, then

$$\alpha = i[(k+1)/s] \quad /i=1,2,\ldots,s-1 \qquad \text{(A)}$$

If $\beta^S-1 \equiv 0 \pmod{k+1}$, then (by Theorem A7.1.12):

$s = (0,k+1) = k+1$ [by (A7.1.1e)], hence by (A): $\alpha = 1,2,\ldots,k$. Hence:

$$\text{If } \beta^S \equiv 1 \pmod{k+1},\text{ any } \alpha = 1,2,\ldots,k \qquad \text{(B)}$$

Consider solutions for β^S-1. From Theorem A7.2.5:

If $(\alpha,k+1) \mathrel{\hat{=}} r$, then

$$\beta^S-1 = i[(k+1)/r] \quad /i=0,1,\ldots,r-1 \qquad \text{(C)}$$

From (7.29a), the first S-1 powers of β must be different than 1 (mod k+1). From Theorem A7.2.9, this can be achieved either if $(\beta,k+1) > 1$, or if $S \le \Phi(k+1)$, in case $(\beta,k+1)=1$.

Let $(\beta,k+1) > 1$: Then, (7.29a) is satisfied, for any β

not relatively prime to k+1 and any S. Let $(\beta,k+1) \hat{=} d > 1$. Then, since $d|\beta \Longrightarrow d|\beta^i$ hence, all S powers of β are multiples of d. These S powers must be incongruent to each other (mod k+1), so that the 1st-column elements $(\alpha,\beta\alpha,\beta^2\alpha,\ldots,\beta^{S-1}\alpha)$ are distinct. Obviously, there are exactly $(k+1)/d$ such multiples, $1,d,2d,[(k+1)/d]d = k+1$, and since the last one must be excluded, β^i can assume no more than $(k+1)/d-1$ distinct values. Then:

$$S \le (k+1)/(k+1,\beta) - 1 \qquad \text{(D)}$$

<u>Let $(\beta,k+1) = 1$</u>: Then, by (A7.2.7b): $\beta^{\Phi(k+1)} \equiv 1 \pmod{k+1}$, hence it is necessary that:

$$S \le \Phi(k+1) \qquad \text{(E)}$$

<u>Let $(\alpha,k+1) = 1$</u>: Then, from (C), $\beta^S \equiv 1 \pmod{k+1}$, hence, from Theorem A7.2.9, $(\beta,k+1) = 1$. Since all first S-1 powers of β must be distinct, it follows from Definition A7.3.1, that:

$$\mathrm{Ord}_{k+1}(\beta) = S \quad /(\beta,k+1) = 1 \qquad \text{(F)}$$

Also, by (A7.3.2c): $$S \mid \Phi(k+1) \qquad \text{(G)}$$

Since $(\beta,k+1) = 1 \Longrightarrow (\beta^i,k+1) = 1$ (by Theorem A7.1.11)

Also, since $(\alpha,k+1) = 1 \Longrightarrow (\alpha\beta^i,k+1) = 1$ [by (A7.1.8)]

So, all 1st-column elements are relatively prime to k+1:

$$(a_{i,1},k+1) = 1 \quad /i=1,2,\ldots,S \qquad \text{(H)}$$

By (H) & Theorem 7.3: $m \le k-1$. Hence:

$$m_{max} = k-1 \qquad \text{(I)}$$

QED

APPENDIX 7.12: <u>CYCLICALLY-DECODABLE TYPE-Bi CODES</u>

A7.12.1. <u>Proof of Theorem 7.27</u>

The (p,J) type-B2 code has parameters $n = (p+1)(J+p-1)$, $k = (p+1)(p-1)$ & $m+1 = p-1$, where $2 \le J \le p-1$ (by Theorem 7.17). Let $r \hat{=} (\alpha,k+1)$. Since $\alpha < k+1 = p^2$, then $r \hat{=} (\alpha,k+1) = (\alpha,p^2) = 1$ or p.* <u>Assume</u> that $(\alpha,k+1) = 1$. Then, from (7.30f):

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

$(n-k) \mid \Phi(k+1) \Longrightarrow [(p+1)(J+p-1) - (p+1)(p-1)] \mid \Phi(p^2)$

$\Longrightarrow J(p+1) \mid p^2(p-1)/p = p(p-1)$ (by Theorem A7.1.15)

Also, by Theorem 7.25: $(n-k) \mid k \Longrightarrow J(p+1) \mid (p+1)(p-1)$

Hence, by Theorem A7.1.6,

$$J(p+1) \mid \bigl(p(p-1),(p+1)(p-1)\bigr) = (p-1)(p,p+1) = p-1$$

it is necessary that J(p+1) divides p-1, which is impossible. Hence, <u>contradiction</u> and $r \triangleq (\alpha,k+1) = p \Longrightarrow \alpha = jp \ /1\le j\le p-1$.

By Theorem 7.25 (iii), any 1st-column element, say, x is congruent to $jp\beta^i$ $/0\le i\le n-k-1$: $x \equiv jp\beta^i \pmod{p^2} \Longrightarrow p \mid p^2 \mid (x-jp\beta^i) \Longrightarrow p \mid x$. Hence, the 1st column contains only multiples of p. Similarly, any element of column, say, z is congruent to zx (where x is the corresponding 1st-column element), or congruent to $z\mu p \pmod{p^2}$, hence a multiple of p. Then, the IA contains only multiples of p, hence there are error bits not checked by syndrome bits. <u>Hence, no type-B2 code is also a type-C code</u>.

According to Theorem 7.18, a (p,q,J) type-B3 code has parameters $n = (q+2)J+pq-1$, $k = pq-1$ & $m+1 = p-1$, where $p<q<2p$ and $2\le J\le(q-1)/2$. Let $r \triangleq (\alpha,k+1) = (\alpha,pq)$. Since $\alpha<k+1=pq$, then r = 1 or p or q. <u>Assume</u> that $(\alpha,k+1) = r = 1$. Then, from (7.30f):

$(n-k) \mid \Phi(k+1) \Longrightarrow [(q+2)J+pq-1 - (pq-1)] \mid \Phi(pq) \Longrightarrow$

$J(q+2) \mid pq(p-1)(q-1)/(pq) = (p-1)(q-1)$ (by Theorem A7.1.15)

Also, by Theorem 7.25: $(n-k) \mid k \Longrightarrow J(q+2) \mid pq-1$

Hence,

$J(q+2) \mid \bigl(pq-1,pq-1-(p+q)\bigr) = (pq-1,p+q)$ (by Theorem A7.1.12)

Hence, $J(q+2) \mid p+q \Longrightarrow q+2 \mid p+q \Longrightarrow$

there exists integer b: $p+q = b(q+2)$ $/b=1,2,..$

If b = 1, then $p+q = q+2 \Longrightarrow p = 2$ but, by Theorem 7.18, p is an odd prime. Hence b>1. Then:

$p+q = bq+2b \Longrightarrow p = (b-1)q+2b > q$, which <u>contradicts</u>

the assumption that $p < q$. Hence, $(a,k+1) = p$ (or q) $\Longrightarrow a = jp$ (μq), where $1 \le j \le p-1$ ($1 \le \mu \le q-1$). Following the same procedure as for the type-B2 codes, one concludes that the IA elements are all multiples of p (or q). Hence, <u>there are no type-B3 codes which are also type-C codes</u>.

Consider the k type-B4 code. By Theorem 7.22, this is a $(k+2,k,k-1)$ code, with k = odd. Since $n-k = 2 \nmid k$, <u>there is no type-B4 code which is also a type-C code</u>.

QED

A7.12.2. <u>Proof of Theorem 7.31</u>

<u>Assume</u> that the equivalent conditions for the existence of a $(k+J,k,k-1)$ type-B self-orthogonal code, which is also a type-C code, (see Theorem 7.30) hold true:

$$\left(a\beta^i, k+1\right) = 1 \quad /\text{for } i=0,1,\ldots,J-1 \qquad \text{(A)}$$

$$a\beta^i \not\equiv a\beta^j \pmod d \quad /d|k+1,\ d>1,\ i,j=0,1,..,J-1\ \&\ i \ne j \qquad \text{(B)}$$

From (A), for $i=0$: $$(a,k+1) = 1 \qquad \text{(C)}$$

Let $e \triangleq (\beta,k+1)$. Then, $e|\beta \Longrightarrow e|a\beta$ and since $e|k+1$, it follows that $e \mid (a\beta,k+1) = 1$, by (A). Hence,

$$(\beta,k+1) = 1 \qquad \text{(D)}$$

Hence: $$\text{(A)} \Longrightarrow (a,k+1) = (\beta,k+1) = 1$$

Also, the <u>converse</u> is true, by Theorems A7.1.11 & A7.1.13:

$$\text{(A)} \Longleftrightarrow (a,k+1) = (\beta,k+1) = 1 \qquad \text{(E)}$$

From (B):

$$a\beta^i \not\equiv a\beta^j \pmod d \Longleftrightarrow d \nmid a\beta^j(\beta^{i-j}-1) \qquad \text{(F)}$$

where, without loss of generality, it has been assumed that $i>j$. <u>Assume</u> that there exists d, such that $d \mid (\beta^{i-j}-1)$. Then, $d \mid a\beta^j(\beta^{i-j}-1)$, which <u>contradicts</u> (B). Hence, (F) $\Longrightarrow d \nmid (\beta^{i-j}-1)$.

<u>Conversely</u>, let $d \nmid (\beta^{i-j}-1)$ & <u>assume</u> that $d \mid a\beta^j(\beta^{i-j}-1)$. Since $(\beta,k+1) = 1$, then $(\beta^z,k+1) = 1$ (from Theorem

A7.1.11). Also, since $(\alpha,k+1) = 1 \Longrightarrow (\alpha\beta^j,k+1) = 1$. Then, if $(\alpha\beta^j,d) \hat{=} e$, since $e|d|(k+1)$ & $e|\alpha\beta^j \Longrightarrow e \mid (\alpha\beta^j,k+1) \Longrightarrow e=1$. Since $(d,\alpha\beta^j) = 1$ and $d \mid \alpha\beta^j(\beta^{i-j}-1) \Longrightarrow d \mid (\beta^{i-j}-1)$, which <u>contradicts</u> the hypothesis, hence $d \nmid \alpha\beta^j(\beta^{i-j}-1)$. Then: $d \nmid (\beta^{i-j}-1) \Longrightarrow$ (B).

So: $$\text{(B)} \Longleftrightarrow d \nmid (\beta^{i-j}-1) \qquad \text{(G)}$$

From (B) & (G), for all $i,j \in [0,J)$ $/i>j$ & $d|k+1$, $d>1$:

$$\alpha\beta^i \not\equiv \alpha\beta^j \pmod d \Longleftrightarrow d \nmid (\beta^{i-j}-1) \qquad \text{(H)}$$

$0 \le j<i<J \Longleftrightarrow i-j>0$ & $i-j<J \Longleftrightarrow 1 \le i-j=z \le J-1$. So:

For all $z=1,2,\ldots,J-1$ & $j=1,2,\ldots,J-z$:

$$\alpha\beta^{z+j} \not\equiv \alpha\beta^j \pmod d \Longleftrightarrow d \nmid (\beta^z-1)$$

$$\Longleftrightarrow \beta^z \not\equiv 1 \pmod d \quad /z=1,2,\ldots,J-1$$

$$\Longleftrightarrow \mathrm{Ord}_d(\beta) \ge J \qquad \text{(I)}$$

So, $$\text{condition (B)} \Longleftrightarrow \mathrm{Ord}_d(\beta) \ge J \qquad \text{(J)}$$

Since the assumption that (A) & (B) hold true imply the existence of a type-C code, then it is necessary that (7.29b) holds true, i.e. that $\beta^J \equiv 1 \pmod{k+1}$. Since $(\beta,k+1) = 1$ and from (I), all the powers of β^j $/j<J$ are different than 1 (mod k+1), it follows that:

$$\mathrm{Ord}_{k+1}(\beta) = J \Longrightarrow \beta^J \equiv 1 \pmod{k+1} \Longleftrightarrow k+1 \mid \beta^J-1$$

and since $d \mid k+1$: $d \mid \beta^J-1 \Longrightarrow \beta^J \equiv 1 \pmod{k+1} \Longrightarrow$

$$\mathrm{Ord}_d(\beta) \le J \qquad \text{(K)}$$

From (J) & (K): $$\text{condition (B)} \Longleftrightarrow \mathrm{Ord}_d(\beta) = J \qquad \text{(L)}$$

So:

$$\text{(A) \& (B)} \Longleftrightarrow (\alpha,k+1)=(\beta,k+1)=1 \ \& \ \mathrm{Ord}_d(\beta) = J \qquad \text{(M)}$$

(M), above is a set of necessary & sufficient conditions for the existence of a self-orthogonal type-B code, which is also a type-C code.

The only <u>condition</u> on J is that it is the order of some integer, say, d. By (A7.3.2c), it is necessary that $J \mid \Phi(d)$. No other J can then be acceptable.

$J \mid \Phi(d)$ for any non-trivial divisor, d, of k+1 (N)

It will be proved that (N) is equivalent to $J \mid \theta(k+1)$.

Let:

$k+1 = p_1^{a(1)} p_2^{a(2)} \cdots p_r^{a(r)} / p_1 < p_2 < \cdots < p_r$ & $a(i) \geq 1$, $i=1,2,..,r$ (O)

<u>Assume</u> that $J \mid \Phi(d)$ $/d>1$ & $d|k+1$. Then, from (O):

$J \mid \Phi(p_i) = p_i - 1$ (by Theorem A7.1.15), for $i=1,2,...,r$.

$\Longrightarrow$ $J \mid (p_1-1, p_2-1, ..., p_r-1) \hat{=} \theta(k+1)$.

<u>Conversely</u>, assume that $J \mid (p_1-1, p_2-1, ..., p_r-1) \hat{=} \theta(k+1)$.

From (O) & Theorem A7.1.3, for any non-trivial divisor, d, of k+1:

$d = p_1^{d(1)} p_2^{d(2)} \cdots p_r^{d(r)} / p_1 < p_2 < \cdots < p_r$ & $d(i) \geq 0$ $i=1,2,..,r$ (P)

From (P) and Theorem A7.1.15:

$\Phi(d) = p_1^{d(1)} p_2^{d(2)} \cdots p_r^{d(r)} (p_1-1)(p_2-1) \cdots (p_r-1)/(p_1 p_2 \cdots p_r)$ (Q)

where if $d(x)=0$, the factor $p_x^{d(x)}(p_x-1)/p_x$ is missing.

$\Longrightarrow$

$\Phi(d) = p_1^{d(1)-1} p_2^{d(2)-1} \cdots p_r^{d(r)-1} (p_1-1)(p_2-1) \cdots (p_r-1)$ $/d(i) \geq 0$ (R)

where if $d(x)=0$, the factor $p_x^{d(x)-1}(p_x-1)$ is missing.

Hence, $\Phi(d)$ is the product of factors $p_x^{d(x)-1}(p_x-1)$, where $d(x)-1 \geq 0$. Hence, for all such factors, $p_x^{d(x)-1} \geq 1$ $\Longleftrightarrow$ $p_x^{d(x)-1}(p_x-1) \geq p_x-1$. Furthermore, for each $d>1$ there exists at least one such factor, because there exists at least one prime factor of d. Since J divides all p_x-1, it follows that it also divides all $\Phi(d)$.

QED

A7.12.3. <u>Proof of Theorem 7.32</u>

Let k+1 be any odd positive integer and J any integer $J \geq 2$, such that $J \mid \theta(k+1)$. It will be proved that ß, given by (7.33a) (p. 218), has order J modulo any non-trivial divisor, d, of k+1.

Consider the prime factorization of k+1:

$$k+1 = \prod_{i=1}^{r} p_i^{a(i)} \quad /a(i)\geq 1, \quad i=1,2,\ldots,r \qquad (A)$$

$$\beta \equiv \sum_{i=1}^{r} g_i^{f(i)/J}\left[(k+1)/p_i^{a(i)}\right]^{f(i)} \pmod{k+1} \qquad (7.33a)$$

where: $$f(i) \triangleq p_i^{a(i)-1}(p_i-1) \quad /i=1,2,\ldots,r \qquad (7.33b)$$

Consider, at first, divisors $p_i^{b(i)}$ $/1\leq b(i)\leq a(i),\ 1\leq i\leq r$ and reduce $\beta \pmod{p_j^{b(j)}}$ $/j\in[1,r]$.

Since, $b(j)-1 \geq 0 \implies p_j^{b(j)-1} \geq 1$. Since, also, $J \mid \theta(k+1) \triangleq (p_1-1,p_2-1,\ldots,p_r-1) \implies$

$$J \mid p_j^{b(j)-1}(p_j-1) \triangleq \tau(j) = f(j)/p_j^{a(j)-b(j)} \qquad (B)$$

For all $i=1,2,\ldots,r$ $/i\neq j$:

$$p_j^{b(j)} \mid (k+1)/p_i^{a(i)} \implies p_j^{b(j)} \mid \left[(k+1)/p_i^{a(i)}\right]^{\tau(i)}$$

$$\implies p_j^{b(j)} \mid g_i^{\tau(i)/J}\left[(k+1)/p_i^{a(i)}\right]^{\tau(i)} \text{ for all } i=1,2,\ldots,r\ /i\neq j \implies$$

$$g_i^{\tau(i)/J}\left[(k+1)/p_i^{a(i)}\right]^{\tau(i)} \equiv 0 \pmod{p_j^{b(j)}} \quad /i=1,2,\ldots,r\ \&\ i\neq j$$

Since $a(i)-b(i) \geq 0$ $/i=1,2,\ldots,r \implies p_i^{a(i)-b(i)} \geq 1$ $/i=1,2,\ldots,r$. If the last congruence is raised to $p_i^{a(i)-b(i)}$ (allowed by Theorem A7.2.2), since $\tau(i)p_i^{a(i)-b(i)} = f(i)$ [by (B)], the following congruence will be obtained:

$$g_i^{f(i)/J}\left[(k+1)/p_i^{a(i)}\right]^{f(i)} \equiv 0 \pmod{p_j^{b(j)}} \quad /i=1,2,\ldots,r\ \&\ i\neq j \implies$$

$$\beta \equiv g_j^{f(j)/J}\left[(k+1)/p_j^{a(j)}\right]^{f(j)} \pmod{p_j^{b(j)}} \quad /\substack{1\leq b(j)\leq a(j) \\ \&\ 1\leq j\leq r} \qquad (C)$$

From (A), $(k+1)/p_j^{a(j)}$ contains no prime factors p_j. Hence,

$$\left((k+1)/p_j^{a(j)}, p_j^{b(j)}\right) = 1 \qquad (D)$$

By Theorem A7.1.15 & (B):

$$\Phi(p_j^{b(j)}) = p_j^{b(j)-1}(p_j-1) \triangleq \tau(j) \qquad (E)$$

By (D) & Theorem A7.2.6:

$$\left[(k+1)/p_j^{a(j)}\right]^{\tau(j)} \equiv 1 \pmod{p_j^{b(j)}} \qquad (F)$$

Raising eqn (F) to $p_j^{a(j)-b(j)}$, noting from (B) that $f(j) =$

$\tau(j)p_j^{a(j)-b(j)}$ and substituting in (C):

$$\beta \equiv g_j^{f(j)/J} \pmod{p_j^{b(j)}} \quad /1\le b(j)\le a(j)\ \&\ 1\le j\le r \qquad (G)$$

If $g_i \hat{=}$ primitive root (mod p_i) such that $g_i^{p_i-1} \not\equiv 1 \pmod{p_i^2}$ then, by Theorem A7.3.4, g_i is also a primitive root modulo $p_i^{c(i)}$, for all $c(i)\ge 1$. So, g_j has order $\Phi(p_j^{b(j)}) \pmod{p_j^{b(j)}}$, or using Theorem A7.1.15:

$$\mathrm{Ord}(g_j) = p_j^{b(j)-1}(p_j-1) \pmod{p_j^{b(j)}} \qquad (H)$$

Then, by Theorem A7.3.1:

$$\mathrm{Ord}\left(g_j^{f(j)/J}\right) = \mathrm{Ord}\left(g_j\right) \Big/ \left(\mathrm{Ord}\left(g_j\right), f(j)/J\right) \quad \left(\mathrm{mod}\ p_j^{b(j)}\right)$$

and using (H) & (E), in (mod $p_j^{b(j)}$):

$$\mathrm{Ord}\left(g_j^{f(j)/J}\right) = \left[p_j^{b(j)-1}(p_j-1)\right] \Big/ \left(p_j^{b(j)-1}(p_j-1), p_j^{a(j)-1}(p_j-1)/J\right) \longrightarrow$$

[multiply numerator & denominator of the RHS, by J]

$$\mathrm{Ord}\left(g_j^{f(j)/J}\right) = J\left[p_j^{b(j)-1}(p_j-1)\right] \Big/ \left(p_j^{b(j)-1}(p_j-1)J, p_j^{a(j)-1}(p_j-1)\right) \longrightarrow$$

$$\mathrm{Ord}\left(g_j^{f(j)/J}\right) = J\left[p_j^{b(j)-1}(p_j-1)\right] \Big/ \left[p_j^{b(j)-1}(p_j-1)(J, p_j^{a(j)-b(j)})\right] \longrightarrow$$

$$\left(\mathrm{Ord}\ g_j^{f(j)/J}\right) = J \,/\, (J, p_j^{a(j)-b(j)}) \qquad (I)$$

Since $J \mid p_i-1$ for all $i=1,2,\dots,r$, it follows that $J < p_j$, hence $(J,p_j) = 1 \longrightarrow (J, p_j^{a(j)-b(j)}) = 1$ (by Theorem A7.1.11). Hence, from (I) & (G):

$$\mathrm{Ord}(\beta) = J \pmod{p_j^{b(j)}} \quad /1\le b(j)\le a(j)\ \&\ 1\le j\le r \qquad (J)$$

Consider, next, any non-trivial divisor d, of k+1 and <u>assume</u> that there exists an integer $x\in[1,J-1]$, such that $\beta^x \equiv 1 \pmod d$. Let p be a prime factor of d. Then $p \mid d \mid \beta^x - 1$, hence $\beta^x \equiv 1 \pmod p$, which <u>contradicts</u> (J). Then:

$$\mathrm{Ord}_d(\beta) \ge J \quad \text{for all non-trivial divisors of } k+1 \qquad (K)$$

Finally, consider again any non-trivial divisor, d, of k+1 and its factorization [from (A)]:

$$d = \prod_{i=1}^{r} p_i^{d(i)} \quad /0\le d(i)\le a(i),\quad i=1,2,\dots,r \qquad (L)$$

From (J): $\beta^J \equiv 1 \pmod{p_i^{a(i)}}$ $/i=1,2,\dots,r$ $\longrightarrow$

$$p_i^{a(i)} \mid \beta^J-1 \quad /i=1,2,\ldots,r \quad \longrightarrow$$

$$p_i^{d(i)} \mid p_i^{a(i)} \mid \beta^J-1 \quad /i=1,2,\ldots,r \quad [\text{by (L)}, \ d(i)\le a(i)] \quad \longrightarrow$$

$$p_i^{d(i)} \mid \beta^J-1 \quad /i=1,2,\ldots,r \qquad (M)$$

Since $\left(p_1^{d(1)},p_2^{d(2)},\ldots,p_r^{d(r)}\right) = 1$, it follows from (M) and Theorem A7.1.14, that:

$$p_1^{d(1)}p_2^{d(2)}\cdots p_r^{d(r)} = d \mid \beta^J - 1 \quad \longrightarrow$$

$$\beta^J \equiv 1 \pmod d \quad \text{for any non-trivial divisor of } k+1 \quad \longrightarrow$$

$$\mathrm{Ord}_d(\beta) \le J \quad \text{for any non-trivial divisor of } k+1 \qquad (N)$$

From (K) & (N):

For any non-trivial divisor of k+1: $\mathrm{Ord}_d(\beta) = J$

QED

A7.12.4. Examples of Type-C5 Codes

Example A7.12.1: Let k+1 = prime = 23. Then $\theta(23) = 22$, and $J\ge2$, $J \mid 22$. Let $J = 11$. Then there exists a (22,11) type C5 code, which is a rate R=2/3 (33,22,21) type-B self-orthogonal cyclically decodable code, with exactly J = 11 syndromes checking on each error bit.

From Lemma 7.4, eqn (7.34a):

$$\beta \equiv g^{k/J} \pmod{k+1} \equiv g^{22/11} \pmod{23} \equiv g^2 \pmod{23}$$

where g is a primitive root (mod 23). From TABLE A7.3.1 (p. 446), g=5. Then, from above, $\beta \equiv 25 \pmod{23} \longrightarrow \beta=2$.

For $\alpha=1$, the IA is:

2	4	6	8	10	12	14	16	18	20	22	1	3	5	7	9	11	13	15	17	19	21
4	8	12	16	20	1	5	9	13	17	21	2	6	10	14	18	22	3	7	11	15	19
8	16	1	9	17	2	10	18	3	11	19	4	12	20	5	13	21	6	14	22	7	15
16	9	2	18	11	4	20	13	6	22	15	8	1	17	10	3	19	12	5	21	14	7
9	18	4	13	22	8	17	3	12	21	7	16	2	11	20	6	15	1	10	19	5	14
18	13	8	3	21	16	11	6	1	19	14	9	4	22	17	12	7	2	20	15	10	5
13	3	16	6	19	9	22	12	2	15	5	18	8	21	11	1	14	4	17	7	20	10
3	6	9	12	15	18	21	1	4	7	10	13	16	19	22	2	5	8	11	14	17	20
6	12	18	1	7	13	19	2	8	14	20	3	9	15	21	4	10	16	22	5	11	17
12	1	13	2	14	3	15	4	16	5	17	6	18	7	19	8	20	9	21	10	22	11
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
*	*	*	*		*		*	*			*	*			*		*				

Note that there are k/J = 22/11 = 2 cosets. The columns of the 1st coset have been marked by an *.

Example A7.12.2: Let $k+1 = p^a = 5^2 = 25$. Then $\theta(25) = 4$, and $J \geq 2$, $J \mid 4$. Let $J = 4$. Then there exists a (24,4) type C5 code, which is a rate R=6/7 (28,24,23) type-B self-orthogonal cyclically decodable code, with exactly J = 4 syndromes checking on each error bit.

From Lemma 7.4, eqn (7.34b):

$$\text{If } f \triangleq p^{a-1}(p-1), \text{ then } \beta \equiv g^{f/J} \pmod{k+1}$$

$$\Longrightarrow \quad f = 5^{2-1}(5-1) = 20, \text{ and } \beta \equiv g^{20/4} \pmod{25} \equiv g^5 \pmod{25}$$

where g is a primitive root (mod 5), such that $g^{p-1} \not\equiv 1 \pmod{p^2}$. From TABLE A7.3.1 (p. 446), g=2, and $g^{p-1} \equiv 2^4 \equiv 16 \pmod{25}$, hence g=2 can be used. From above, $\beta \equiv 32 \pmod{25}$ $\Longrightarrow$ $\beta=7$. For $\alpha=1$, the IA is:

7	14	21	3	10	17	24	6	13	20	2	9	16	23	5	12	19	1	8	15	22	4	11	18
24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
18	11	4	22	15	8	1	19	12	5	23	16	9	2	20	13	6	24	17	10	3	21	14	7
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	2	3	3	4	5	1	5	6	4	2	6	6	2	4	6	5	1	5	4	3	3	2	1

Note that there are $k/J = 24/4 = 6$ cosets.

∎

Example A7.12.3: Let $k+1 = p_1 p_2 = 5 \times 13 = 65$. Then $\theta(65) = (4,12) = 4$, and $J \geq 2$, $J \mid 4$. Let $J = 4$. Then there exists a (64,4) type C5 code, which is a rate R=16/17 (68,64,63) type-B self-orthogonal cyclically decodable code, with exactly J = 4 syndromes checking on each error bit.

From Lemma 7.4, eqn (7.34c):

$$\beta \equiv g_1^{(p_1-1)/J} p_2^{p_1-1} + g_2^{(p_2-1)/J} p_1^{p_2-1} \pmod{k+1}$$

where g_1 is a primitive root (mod 5) and g_2 is a primitive root (mod 13). From TABLE A7.3.1 (p. 446), $g_1=2$ & $g_2=2$. Then, from above,

$$\beta \equiv 2^{(5-1)/4} \times 13^{5-1} + 2^{(13-1)/4} \times 5^{13-1} \pmod{65} \quad \Longrightarrow$$

$$\beta \equiv 2 \times 13^4 + 2^3 \times 5^{12} \equiv 1{,}953{,}182{,}122 \pmod{65} \quad \Longrightarrow \quad \beta = 47$$

For $\alpha=1$, the IA is:

47	29	11	58	40	22	4	51	33	15	62	44	26	8	55	37	19	1	48	30	12	59	41	23
64	63	62	61	60	59	58	57	56	55	54	53	52	51	50	49	48	47	46	45	44	43	42	41
18	36	54	7	25	43	61	14	32	50	3	21	39	57	10	28	46	64	17	35	53	6	24	42
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	2	3	4	5	6	4	7	8	9	3	10	11	7	9	12	13	1	13	14	10	6	15	15

5	52	34	16	63	45	27	9	56	38	20	2	49	31	13	60	42	24	6	53	35	17	64	46
40	39	38	37	36	35	34	33	32	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17
60	13	31	49	2	20	38	56	9	27	45	63	16	34	52	5	23	41	59	12	30	48	1	19
25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
5	11	16	12	2	14	16	8	8	16	14	2	12	16	11	5	15	15	6	10	14	13	1	13

28	10	57	39	21	3	50	32	14	61	43	25	7	54	36	18
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
37	55	8	26	44	62	15	33	51	4	22	40	58	11	29	47
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
12	9	7	11	10	3	9	8	7	4	6	5	4	3	2	1

Note that this IA has k/J = 64/4 = 16 cosets.

∎

<u>Example A7.12.4</u>: Let k+1 = 19. Then $\theta(19)$ = 18, and J = 2, 3, 6, 9 & 18. Let J = 6. Then there exists a (18,6) type C5 code, which is a rate R=3/4 (24,18,17) type-B self-orthogonal cyclically decodable code, with exactly J = 6 syndromes checking on each error bit. Its β can be calculated from eqn (7.34a): β = 8. Then, for α=1, the IA is:

8	16	5	13	2	10	18	7	15	4	12	1	9	17	6	14	3	11
7	14	2	9	16	4	11	18	6	13	1	8	15	3	10	17	5	12
18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
11	3	14	6	17	9	1	12	4	15	7	18	10	2	13	5	16	8
12	5	17	10	3	15	8	1	13	6	18	11	4	16	9	2	14	7
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	2	3	2	3	1	1	3	3	1	1	3	2	3	2	2	1

The syndrome register (SYRE) is an (n-k) x (m+1) store of syndrome bits (see Fig. 7.1, p. 210). In this case the SYRE has dimensions 6 x 18. Using Theorem 7.1 and the above IA, one may deduce the co-ordinates (x,y) /$1 \le x \le 6$ & $1 \le y \le 18$, of the syndrome bits that check on each error bit. For instance, since $a_{2,4}$ = 9, $s^{(2)}_{4-1}$ checks on $e^{(9)}_0$. This syndrome bit is in row 2, column 15. In general $s^{(z)}_{w-1}$ is in stage (z,k+1-w). Since, by Theorem 7.1, $s^{(z)}_{w-1}$ checks on $e^{\{a[z,w]\}}_0$:

The syndrome bits checking on $e^{\{a[z,w]\}}_0$, are in stages {z,k+1-w} of the SYRE. (A7.12.1)

From statement (A7.12.1) & the IA, one may deduce the syndrome bits checking on the error bits of the 1st coset:

For $e_0^{(8)}$, $a_{z,w} = 8$, for z=1,2,3,4,5,6 & w=1,12,11,18,7,8, respectively, hence k+1-w=19-w=18,7,8,1,12,11:

$e_0^{(08)}$ is checked by (1,18) (2, 7) (3, 8) (4, 1) (5,12) & (6,11)

$e_0^{(07)}$ is checked by (1,11) (2,18) (3, 7) (4, 8) (5, 1) & (6,12)

$e_0^{(18)}$ is checked by (1,12) (2,11) (3,18) (4, 7) (5, 8) & (6, 1)

$e_0^{(11)}$ is checked by (1, 1) (2,12) (3,11) (4,18) (5, 7) & (6, 8)

$e_0^{(12)}$ is checked by (1, 8) (2, 1) (3,12) (4,11) (5,18) & (6, 7)

$e_0^{(01)}$ is checked by (1, 7) (2, 8) (3, 1) (4,12) (5,11) & (6,18)

If the above SYRE co-ordinates are *rearranged*, the cyclic nature of decoding will become obvious:

$e_0^{(08)}$ is checked by (4, 1) (2, 7) (3, 8) (6,11) (5,12) & (1,18)

$e_0^{(07)}$ is checked by (5, 1) (3, 7) (4, 8) (1,11) (6,12) & (2,18)

$e_0^{(18)}$ is checked by (6, 1) (4, 7) (5, 8) (2,11) (1,12) & (3,18)

$e_0^{(11)}$ is checked by (1, 1) (5, 7) (6, 8) (3,11) (2,12) & (4,18)

$e_0^{(12)}$ is checked by (2, 1) (6, 7) (1, 8) (4,11) (3,12) & (5,18)

$e_0^{(01)}$ is checked by (3, 1) (1, 7) (2, 8) (5,11) (4,12) & (6,18)

Hence, if the SYRE stages for $e_0^{(8)}$ are connected to the MG of the first coset, an upward uniform cyclic shift by one step will allow the decoding of $e_0^{(7)}$, the next shift will decode $e_0^{(18)}$, etc. The syndrome connections to the other two MGs can be similarly deduced:

For the 2nd coset:

$e_0^{(16)}$ is checked by (4, 2) (6, 3) (5, 5) (2,14) (3,16) & (1,17)

$e_0^{(14)}$ is checked by (5, 2) (1, 3) (6, 5) (3,14) (4,16) & (2,17)

$e_0^{(17)}$ is checked by (6, 2) (2, 3) (1, 5) (4,14) (5,16) & (3,17)

$e_0^{(03)}$ is checked by (1, 2) (3, 3) (2, 5) (5,14) (6,16) & (4,17)

$e_0^{(05)}$ is checked by (2, 2) (4, 3) (3, 5) (6,14) (1,16) & (5,17)

$e_0^{(02)}$ is checked by (3, 2) (5, 3) (4, 5) (1,14) (2,16) & (6,17)

For the 3rd coset:

$e_0^{(13)}$ is checked by (4, 4) (6, 6) (2, 9) (5,10) (3,13) & (1,15)

$e_0^{(09)}$ is checked by (5, 4) (1, 6) (3, 9) (6,10) (4,13) & (2,15)

$e_0^{(15)}$ is checked by (6, 4) (2, 6) (4, 9) (1,10) (5,13) & (3,15)

$e_0^{(06)}$ is checked by (1, 4) (3, 6) (5, 9) (2,10) (6,13) & (4,15)

$e_0^{(10)}$ is checked by (2, 4) (4, 6) (6, 9) (3,10) (1,13) & (5,15)

$e_0^{(04)}$ is checked by (3, 4) (5, 6) (1, 9) (4,10) (2,13) & (6,15)

Hence, the *connections to the MGs* are as following:

1st MG: (4, 1) (2, 7) (3, 8) (6,11) (5,12) & (1,18)

2nd MG: (4, 2) (6, 3) (5, 5) (2,14) (3,16) & (1,17)

3rd MG: (4, 4) (6, 6) (2, 9) (5,10) (3,13) & (1,15)

Notice that *the SYRE columns connected to the three gates* are:

<u>For MG1</u>:	1,					7,	8,			11,	12,							18
<u>For MG2</u>:		2,	3,		5,								14,		16,	17		
<u>For MG3</u>:				4,		6,		9,	10,			13,		15				

i.e. there is no overlapping. Furthermore, the sequence of *the bits decoded by each gate* is:

From MG1:	$e_0^{(08)}$	$e_0^{(07)}$	$e_0^{(18)}$	$e_0^{(11)}$	$e_0^{(12)}$	$e_0^{(01)}$
From MG2:	$e_0^{(16)}$	$e_0^{(14)}$	$e_0^{(17)}$	$e_0^{(03)}$	$e_0^{(05)}$	$e_0^{(02)}$
From MG3:	$e_0^{(13)}$	$e_0^{(09)}$	$e_0^{(15)}$	$e_0^{(06)}$	$e_0^{(10)}$	$e_0^{(04)}$

∎

APPENDIX 7.13: <u>INTRODUCTION TO QUADRATIC RESIDUES</u>

This appendix is drawn mainly from Chapter 9 of T.M. Apostol's "Introduction to Analytic Number Theory" [44].

Unless otherwise stated, small latin & greek letters denote integers.

<u>Definition A7.13.1</u>: Let p be any odd prime and $n \not\equiv 0$ (mod p). If congruence $x^2 \equiv n$ (mod p) has a solution it is said that n is a *quadratic residue modulo p* and this is denoted by nRp. If the congruence has no solution it is said that n is a *quadratic nonresidue modulo p* and this is denoted by $n\bar{R}p$.

∎

<u>Definition A7.13.2</u>: Let p be any odd prime. Then, for any n, *Legendre's symbol* (n|p) is defined as following:

$$(n|p) = \begin{cases} 0 & \text{if } n \equiv 0 \pmod p \\ 1 & \text{if } n \not\equiv 0 \pmod p \ \&\ nRp \\ -1 & \text{if } n \not\equiv 0 \pmod p \ \&\ n\bar{R}p \end{cases} \qquad \text{(A7.13.1)}$$

∎

<u>Theorem A7.13.1</u>: *Euler's criterion*: For any odd prime p and any n:

$$(n|p) \equiv n^{(p-1)/2} \pmod p \qquad \text{(A7.13.2)}$$

<u>Proof</u>: See Apostol [44], p. 180.

∎

<u>Theorem A7.13.2</u>: For any odd prime p and any m & n:

$$(mn|p) = (m|p)(n(|p) \qquad \text{(A7.13.3)}$$

<u>Proof</u>: See Apostol [44], pp. 180-1.

∎

<u>Theorem A7.13.3</u>: For any odd prime p, every reduced residue system modulo p contains exactly (p-1)/2 quadratic residues and exactly (p-1)/2 quadratic nonresidues, modulo p. The quadratic residues are congruent to i^2 (mod p) /i=1,2,...,(p-1)/2.

<u>Proof</u>: See Apostol [44], p. 179.

∎

APPENDIX 7.14: PROPERTIES OF THE INITIAL ARRAY

A7.14.1. Proof of Theorem 7.33

Let k+1 be an odd integer with prime decomposition:

$$k+1 = \prod_{i=1}^{r} p_i^{a(i)} \quad /p_1<p_2<\cdots<p_r \quad \& \quad a(i) \geq 1,\ i=1,2,..,r \qquad \text{(A)}$$

Let ß be given by (7.33a). From the proof of Theorem 7.32, in Appendix 7.12 (§ A7.12.3., p. 490) [eqn (G)]:

$$\beta \equiv g_j^{f(j)/J} \quad (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \qquad \text{(B)}$$

where $$f(j) \mathrel{\hat{=}} p_j^{a(j)-1}(p_j-1) \quad /j=1,2,\ldots,r \qquad \text{(C)}$$

and $g_j \mathrel{\hat{=}}$ primitive root (mod p_j), such that $g_j^{p_j-1} \not\equiv 1$ (mod p_j^2) for j=1,2,...,r.

From (B):

$$\beta^{J/2} \equiv g_j^{f(j)/2} \quad (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \qquad \text{(D)}$$

From eqn (J) of § A7.12.3., the order of ß (mod $p_j^{a(j)}$) is J for all j=1,2,...,r. Then:

$$(\beta^{J/2})^2 \equiv 1 \ (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \qquad \text{(E)}$$

One solution of (E), for $\beta^{J/2}$, is $-1 \equiv p_j^{a(j)}-1$ (mod $p_j^{a(j)}$) [+1 is not a solution, because the order of ß is J, hence $\beta^{J/2} \not\equiv 1$ (mod $p_j^{a(j)}$)]. Then, there is no other solution,* for $\beta^{J/2}$, in the range $[1,p_j^{a(j)}]$, hence:

$$\beta^{J/2} \equiv p_j^{a(j)}-1 \ (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \qquad \text{(F)}$$

(F) is a system of congruences with moduli relatively prime in pairs (the unknown is $\beta^{J/2}$). According to the Chinese remainder theorem (Theorem A7.2.10), system (F) has exactly one solution modulo the product of the moduli, i.e. modulo k+1 [see (A)]. Hence there is a unique number, in [1,k+1], which satisfies (F), for all j=1,2,...,r.

From (A), $p_j^{a(j)} \mid k+1$, for all i=1,2,...,r. Then:

$$k \equiv -1 \ (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \quad \Longrightarrow$$

$$k \equiv p_j^{a(j)}-1 \ (\text{mod } p_j^{a(j)}) \quad /j=1,2,\ldots,r \quad \Longrightarrow \quad \text{[from (F)]}$$

* See Vinogradov [46], p. 92.

$\beta^{J/2} \equiv k \pmod{p_j^{a(j)}}$ /j=1,2,...,r $\longrightarrow$

$p_j^{a(j)} \mid \beta^{J/2}-k$ /j=1,2,...,r $\longrightarrow$

$p_1^{a(1)}p_2^{a(2)}\cdots p_r^{a(r)} = k+1 \mid \beta^{J/2}-k$ (by Theorem A7.1.14) $\longrightarrow$

$\beta^{J/2} \equiv k \pmod{k+1}$

QED

A7.14.2. Proof of Theorem 7.34

From Theorem 7.31 (p. 218), the elements of the first column of the IA of the (k,J) type-C5 code, are $a_{x,1} \equiv \alpha\beta^x \pmod{k+1}$, for x=1,2,...,J. For α=1, and since $\mathrm{Ord}_{k+1}(\beta) = J$, $a_{J,1} = 1$ and hence $a_{J,z} \equiv z \pmod{k+1}$, for z=1,2,...,k. Hence,

$$a_{J,z} = z \quad /z=1,2,\ldots,k \qquad \text{(A)}$$

From (7.36) for J=even, $\beta^{J/2} \equiv k \pmod{k+1}$. For α=1:

$\beta^{J/2} \equiv a_{J/2,1} \equiv k \pmod{k+1}$ $\longrightarrow$ $a_{J/2,1} = k \pmod{k+1}$ $\longrightarrow$

$a_{J/2,z} \equiv za_{J/2,1} \equiv zk \pmod{k+1}$ /z=1,2,...,k $\longrightarrow$

$a_{J/2,z} \equiv z(-1) \pmod{k+1}$ /z=1,2,...,k $\longrightarrow$

$a_{J/2,z} \equiv k+1-z \pmod{k+1}$ /z=1,2,...,k $\longrightarrow$

$$a_{J/2,z} = k+1-z \quad /z=1,2,\ldots,k \qquad \text{(B)}$$

For J=even, for all x=1,2,...,J/2 & z=1,2,...,k:

$a_{x,z} + a_{x+J/2,z} \equiv z\beta^x + z\beta^{x+J/2} \pmod{k+1}$ $\longrightarrow$

$a_{x,z} + a_{x+J/2,z} \equiv z\beta^x(1+\beta^{J/2}) \pmod{k+1}$ $\longrightarrow$

$a_{x,z} + a_{x+J/2,z} \equiv z\beta^x(1+k) \pmod{k+1}$ [by (7.36)] $\longrightarrow$

$a_{x,z} + a_{x+J/2,z} \equiv 0 \pmod{k+1}$ $\longrightarrow$

$a_{x,z} + a_{x+J/2,z} = q(k+1)$ /q=integer $\longrightarrow$

Since, $0 < a_{x,z} + a_{x+J/2,z} < 2(k+1)$ $\longrightarrow$ $q = 1$ $\longrightarrow$

$a_{x,z} + a_{x+J/2,z} = k+1$ /x=1,2,...,J/2 & z=1,2,...,k

From the last result, by summing over all x=1,2,...,J/2, (7.37d) follows.

QED

A7.14.3. Proof of Theorem 7.35

From Theorem 7.31 (p. 218), $a_{x,1} \equiv \alpha\beta^x$ (mod k+1), for x=1,2,...,J, where $(\alpha,k+1) = 1$, and $Ord_d(\beta) = J$ for all non-trivial divisors, d, of k+1.* For $\alpha = 1$, and from Definition 7.2 (p. 185):

$$\sum_{x=1}^{J} a_{x,z} \equiv z\sum_{x=1}^{J} a_{x,1} \pmod{k+1} \quad /z=1,2,\ldots,k \qquad \text{(A)}$$

Also,

$$\sum_{x=1}^{J} a_{x,1} \equiv \sum_{x=1}^{J} \beta^x \equiv \left[(\beta^{J+1}-1)/(\beta-1) - 1\right] \pmod{k+1} \quad \longrightarrow$$

$$\sum_{x=1}^{J} a_{x,1} \equiv \beta(\beta^J-1)/(\beta-1) \pmod{k+1} \qquad \text{(B)}$$

Since $(\beta,\beta-1) = 1$ (by Theorem A7.1.6), and $\beta-1 \mid \beta(\beta^J-1)$, then (by Theorem A7.1.10):

$$\beta-1 \mid \beta^J-1 \qquad \text{(C)}$$

Since $Ord_d(\beta) = J$, for $d \mid k+1$, then $\beta^J-1 \equiv 0$ (mod k+1). Hence:

$$k+1 \mid \beta^J-1 \qquad \text{(D)}$$

Let:

$$k+1 = \prod_{i=1}^{r} p_i^{a(i)} \quad /p_1<p_2<\cdots<p_r \quad \& \quad a(i) \geq 1,\ i=1,2,..,r \qquad \text{(E)}$$

Then: $$p_i^{a(i)} \mid k+1 \mid \beta^J-1 \quad /i=1,2,\ldots,r \qquad \text{(F)}$$

Assume that there exists $j\in[1,r]$ such that $(p_j^{a(j)},\beta-1) \triangleq f > 1$. Since $f \mid p_j^{a(j)}$ & $f>1$, then there exists $b\in[1,a(j)]$ such that $f = p_j^b$. Since $f \mid \beta-1$, then $\beta \equiv 1$ (mod p_j^b), which contradicts Theorem 7.31 [that the order of β (mod d), for any $d \mid k+1$, d>1, is J]. Hence,

$$\left(p_i^{a(i)},\beta-1\right) = 1 \quad /i=1,2,\ldots,r \qquad \text{(G)}$$

From (G) & Theorem A7.1.13:

$$\left(p_1^{a(1)}p_2^{a(2)}\cdots p_r^{a(r)},\ \beta-1\right) = \left(k+1,\beta-1\right) = 1 \qquad \text{(H)}$$

* Remember that (a,b) denotes the *greatest common divisor* of a & b.

From (C) & (D), $\beta-1$ & $k+1$ both divide β^J-1, while from (H) they are relatively prime. Then, by Theorem A7.1.14:

$$(\beta-1)(k+1) \mid (\beta^J-1) \implies (k+1) \mid (\beta^J-1)/(\beta-1) \implies$$

$(\beta^J-1)/(\beta-1) \equiv 0 \pmod{k+1}$ and substituting in (B), and then (A):

$$\sum_{x=1}^{J} a_{x,z} \equiv 0 \pmod{k+1} \quad /z=1,2,\ldots,k$$

This proves (7.38a). To prove (7.38b) & (7.38c):

For all $x=1,2,\ldots,J$ & $z=1,2,\ldots,k$:

$$a_{x,z} + a_{x,k+1-z} \equiv z\beta^x + (k+1-z)\beta^x \pmod{k+1} \implies$$

$$a_{x,z} + a_{x,k+1-z} \equiv (k+1+z-z)\beta^x \equiv 0 \pmod{k+1} \implies$$

$$a_{x,z} + a_{x,k+1-z} = q(k+1) \quad /q=\text{integer} \implies$$

Since, $0 < a_{x,z} + a_{x,k+1-z} < 2(k+1) \implies q = 1 \implies$

$$a_{x,z} + a_{x,k+1-z} = k+1 \quad /x=1,2,\ldots,J \ \& \ z=1,2,\ldots,k \implies$$

$$\sum_{x=1}^{J} a_{x,z} + \sum_{x=1}^{J} a_{x,k+1-z} = J(k+1) \quad /z=1,2,\ldots,k$$

QED

A7.14.4. Proof of Theorem 7.36

Let the (k,J) type-C5 code, with J = odd. Let, also, the prime factorization of k+1:

$$k+1 = \prod_{i=1}^{r} p_i^{a(i)} \quad /p_1<p_2<\cdots<p_r \ \ \& \ \ a(i) \geq 1,\ i=1,2,..,r \qquad \text{(A)}$$

Since, by Theorem 7.31, $J \mid \theta(k+1) \triangleq (p_1-1,p_2-1,..,p_r-1)$, then $p_i-1 = q_iJ$, for $i=1,2,\ldots,r$. Since $p_i-1 = \text{even} = q_iJ$ and $J = \text{odd}$, then $q_i = \text{even}$. Hence:

$$(p_i-1)/J = \text{even} \quad /i=1,2,\ldots,r \qquad \text{(B)}$$

From eqn (G) of § A7.12.3. (p. 490):

$$\beta \equiv g_i^{f(i)/J} \pmod{p_i} \quad /i=1,2,\ldots,r \qquad \text{(C)}$$

where: $f(i) \triangleq p_i^{a(i)-1}(p_i-1)$ & $g_i \triangleq$ primitive root $\pmod{p_i}$,

$i=1,2,\ldots,r$.

From (C):

$$\beta^{(p_i-1)/2} \equiv g_i^{[f(i)/J][(p_i-1)/2]} \pmod{p_i} \quad /i=1,2,\ldots,r \qquad (D)$$

Since, from (B), $(p_i-1)/J$ = even $/i=1,2,\ldots,r$, then $(2J) \mid (p_i-1)$, hence $(2J) \mid p_i^{a(i)-1}(p_i-1) \hat{=} f(i)$. From (D):

$\beta^{(p_i-1)/2} \equiv \left[g_i^{f(i)/(2J)}\right]^{p_i-1} \pmod{p_i} \quad /i=1,2,\ldots,r \qquad \Longrightarrow$

$\beta^{(p_i-1)/2} \equiv 1 \pmod{p_i} \quad /i=1,2,\ldots,r$ (by Theorem A7.2.6)

$\Longrightarrow \quad (\beta|p_i) \equiv 1 \pmod{p_i} \quad /i=1,2,..,r$ (by Theorem A7.13.1)

$\Longrightarrow \quad (\beta|p_i) = 1 \pmod{p_i} \quad /i=1,2,.,r$ (by Definition A7.13.2)

$\Longrightarrow \quad (\beta^2|p_i) = 1 \pmod{p_i} \quad /i=1,2,..,r$ (by Theorem A7.13.2)

$\Longrightarrow \quad (\beta^3|p_i) = 1 \pmod{p_i} \quad /i=1,2,..,r$ (by Theorem A7.13.2)

etc

$\Longrightarrow \quad (\beta^x|p_i) = 1 \pmod{p_i} \quad /i=1,2,..,r \ \& \ x=1,2,..,J \qquad (E)$

Since $a_{x,1} \equiv \beta^x \pmod{k+1} \quad /x=1,2,\ldots,J$:

$\Longrightarrow \quad k+1 \mid a_{x,1} - \beta^x \quad /x=1,2,\ldots,J$

$\Longrightarrow \quad p_i \mid k+1 \mid a_{x,1} - \beta^x \quad /x=1,2,\ldots,J \ \& \ i=1,2,\ldots,r$

$\Longrightarrow \quad a_{x,1} \equiv \beta^x \pmod{p_i} \quad /x=1,2,..,J \ \& \ i=1,2,..,r \qquad (F)$

From Euler's criterion (Theorem A7.13.1),

If $n \equiv m \pmod{p} \quad \Longrightarrow$

$(n|p) \equiv n^{(p-1)/2} \equiv m^{(p-1)/2} \equiv (m|p) \pmod{p}$

and since (by Definition A7.13.2), $(k|p) = 0$, 1, or $p-1$:

$$\text{If } n \equiv m \pmod{p} \quad \Longrightarrow \quad (n|p) = (m|p) \qquad (A7.14.1)$$

Then, applying (A7.14.1) to (E) & (F):

$$(a_{x,1}|p_i) = 1 \pmod{p_i} \quad /x=1,2,\ldots,J \ \& \ i=1,2,..,r \qquad (G)$$

Hence, the elements of the first column are all quadratic residues modulo any prime factor of $k+1$. Furthermore, no two such quadratic residues (mod p_i) are congruent to each other (mod p_i), because then there will exist $x,y \in [1,J]$ $/x>y$ such that $\beta^x \equiv \beta^y \pmod{p_i}$, hence $\beta^{x-y} \equiv 1 \pmod{p_i}$ [because $(\beta^y,p_i) = 1$ - see Theorem A7.2.4], and since x-y<J this contradicts Theorem 7.31 [Ord(β) = J (mod p_i)].

For all x=1,2,...,J & z=1,2,...,k, since

$a_{x,z} \equiv za_{x,1} \pmod{k+1} \Longrightarrow a_{x,z} \equiv za_{x,1} \pmod{p_i}$

and using (A7.14.1):

$(a_{x,z}|p_i) = (za_{x,1}|p_i) \pmod{p_i}$ /x=1,2,...,J & i=1,2,...,r

while from Theorem A7.13.2 & (G):

$(a_{x,z}|p_i) = (z|p_i) \pmod{p_i}$ /x=1,2,..,J & i=1,2,..,r (H)

Hence, for every prime factor, p_i, of k+1, a column of the IA contains either multiples of p_i, or quadratic residues, or quadratic nonresidues (mod p_i). The first column contains always quadratic residues.

QED

Example A7.14.1: Let k+1 = 11 x 31 = 341. Then, $\theta(341) = (10,30) = 10$, hence J = 2,5 & 10. Let J=5. Then there exists a (340,5) type-C5 code (by Theorem 7.31), which is a (345,340,339) type-B SO cyclically decodable code with exactly 5 syndromes checking on each error bit. From Lemma 7.4 (p. 219), since k+1 = p_1p_2, and since, from TABLE A7.3.1, $g_1=2$ & $g_2=3$:

$\beta \equiv g_1^{10/5} \times 31^{10} + g_2^{30/5} \times 11^{30} \pmod{341} \Longrightarrow$

$\beta \equiv 2^2 \times 31^{10} + 3^6 \times 11^{30} \pmod{341} \Longrightarrow$

$\beta \equiv 4 \times (31^5)^2 + 729 \times (11^6)^5 \pmod{341} \Longrightarrow$

$\beta \equiv 4 \times 155^2 + 729 \times 66^5 \pmod{341} \Longrightarrow$

$\beta \equiv 4 \times 155 + 729 \times 187 \pmod{341} \Longrightarrow$

$\beta \equiv 136{,}943 \equiv 202 \pmod{341}$

Then, the elements of the zth column are $z \times 202^i$ /i=1,2,...,5, where $1 \le z \le 340$.

Column 1: 202, 225, 97, 157, 1 C(1) = 682 = 341 x 2

Column 2: 63, 109, 194, 314, 2 C(2) = 682 = 341 x 2

Column 3: 265, 334, 291, 130, 3 C(3) = 1,023 = 341 x 3

Column 4: 126, 218, 47, 287, 4 C(4) = 682 = 341 x 2

etc

Column 11: 176, 88, 44, 22, 11 C(11) = 341

etc

Column 62: 248, 310, 217, 186, 62 C(62) = 1,023 = 341 x 3

etc

The quadratic residues (mod p) are given by i^2 (mod p), for i=1,2,...,(p-1)/2 (see Theorem A7.13.3):

(n|11) = 1, for n = 1,3,4,5,9

(n|31) = 1, for n = 1,2,4,5,7,8,9,10,14,16,18,19,20,25,28

If the elements of the 1st column are reduced:

Mod 11: 4,5,9,3,1 [all quadratic residues (mod 11)]

Mod 31: 16,8,4,2,1 [all quadratic residues (mod 31)]

Since (3|11) = (4|11) = 1, columns 3 & 4 are expected to be made of quadratic residues (mod 11), while column 2 should be made of quadratic nonresidues (mod 11). Since 62 ≡ 7 (mod 11), then (62|11) = -1, hence column 62 should be made of quadratic nonresidues.

Column 2 (mod 11): 8,10,7,6,2 [q. nonresidues (mod 11)]

Column 3 (mod 11): 1,4,5,9,3 [q. residues (mod 11)]

Column 4 (mod 11): 5,9,3,1,4 [q. residues (mod 11)]

Column 62 (mod 11): 6,2,8,10,7 [q. nonresidues (mod 11)]

Since (2|31) = (4|31) = 1, columns 2 & 4 are expected to be made of quadratic residues (mod 31), while columns 3 & 11 should be made of quadratic nonresidues (mod 31):

Column 2 (mod 31): 1,16,8,4,2 [q. residues (mod 31)]

Column 3 (mod 31): 17,24,12,6,3 [q. nonresidues (mod 31)]

Column 4 (mod 31): 2,1,16,8,4 [q. residues (mod 31)]

Column 11 (mod 31): 21,26,13,22,11 [q. nonresidues (mod 31)]

Since $(11|11) = 0$, column 11 is expected to contain only multiples of 11 ($176 = 16 \times 11$, etc). Since $(62|31) = 0$, column 62 is expected to contain only multiples of 31 ($248 = 8 \times 31$, $217 = 7 \times 31$, $186 = 6 \times 31$, etc).

APPENDIX 7.15: EFFECTIVE CONSTRAINT-LENGTH

A7.15.1. Proof of Theorem 7.39

From eqn (7.38a) (p. 222) (see also Theorem 7.38), there exists integer q(z), such that:

$$\sum_{x=1}^{J} a_{x,z} = (k+1)q(z) \quad /z=1,2,\ldots,k \qquad (A)$$

Note: Unless otherwise stated, MAX & MIN will be assumed over all z=1,2,...,k.

By Theorem 7.39 (p. 224) & Definition 5.9 (p. 145):

$$n_E = \mathrm{MAX}\left\{1 + \sum_{x=1}^{J} a_{x,z}\right\} = \mathrm{MAX}\left\{1+(k+1)q(z)\right\} \quad \text{[by (A)]} \longrightarrow$$

$$n_E = 1 + (k+1)\mathrm{MAX}\{q(z)\} \qquad (B)$$

From (A) & (7.38c):

$$(k+1)q(z) + (k+1)q(k+1-z) = J(k+1) \quad /z=1,2,\ldots,k \longrightarrow$$

$$q(z) + q(k+1-z) = J \quad /z=1,2,\ldots,k \qquad (A7.15.1)$$

From (B) & (A7.15.1):

$$n_E = 1 + (k+1)\mathrm{MAX}\{q(z)\} = 1 + (k+1)\mathrm{MAX}\{J - q(k+1-z)\}$$

As z takes on values 1,2,...,k, so does k+1-z. Then:

$$n_E = 1 + (k+1)\mathrm{MAX}\{q(z)\} = 1 + (k+1)\left[J - \mathrm{MIN}\{q(z)\}\right] \qquad (C)$$

(C) is the first result of the theorem.

From (A7.15.1): $1 \le q(z) \le J-1 \quad /z=1,2,...,k$ (A7.15.2)

because q(z) cannot be negative or zero, as the quotient of two positive numbers, hence $q(z) \ge 1$. Also, if $q(z) = 1$ then there exists an IA column, such that $q(w) = J-1$ [$w = k+1-z$ - by (A7.15.1)].

Consider bounds for MAX{q(z)}:

<u>Assume</u> that $MAX\{q(z)\} = q(w) < (J+1)/2$. Then, by (A7.15.1):

$q(k+1-w) = J - q(w) > J - (J+1)/2 = (J-1)/2 \Longrightarrow$

$q(k+1-w) \ge (J+1)/2 > q(w) \Longrightarrow$ <u>contradiction</u>. Hence:

$MAX\{q(z)\} \ge (J+1)/2$. Also, from (A7.15.2): $MAX\{q(z)\} \le J-1$

So: $(J+1)/2 \le MAX\{q(z)\} \le J-1$ (D)

From (C) & (D), the 2nd result follows.

Note, from (D), that for J = 3: $2 \le MAX\{q(z)\} \le 2 \Longrightarrow MAX\{q(z)\} = 2$.

QED

A7.15.2. <u>Proof of Theorem 7.40</u>

Let a (p-1,(p-1)/2) type-C5 code, where p is any odd prime, and $p \equiv 3 \pmod 4$. This code has $J = (p-1)/2$, and because $p = 4q+3$ (q=integer), $(p-1)/2 = J = 2q+1$ = odd. By Theorem 7.31, (p-1)/2 must divide p-1 (which it does). By Theorem 7.36, the first column of the IA is made of $J = (p-1)/2$ distinct quadratic residues modulo any prime factor of k+1 and, by Theorem A7.13.3, there are exactly (p-1)/2 quadratic residues (mod p). Hence, the first column contains all the quadratic residues (mod p). The elements of the IA are reduced (mod k+1) = (mod p), in the range [1,k] = [1,p].

The code IA is made of $k/J = (p-1)/[(p-1)/2] = 2$ cosets. The first coset contains all the quadratic residues (mod p), hence the 2nd coset contains all the quadratic nonresidues (mod p). Hence, by Theorems 7.39 & 7.37, n_E-1 equals the sum of the quadratic residues, or the sum of the quadratic non-residues, whichever is greater*.

It has been observed (and it supposed by the author that

* All reduced (mod p) in the range [1,p].

it has been proved indirectly) that "...for $p \equiv 3 \pmod 4$, there are always more quadratic residues than nonresidues in the first half of the range from 0 to p. Again, no direct proof is known" (see H. Davenport [48], p. 9).

Then, from the above, there are more quadratic residues, than nonresidues in $[1,(p-1)/2] = [1,J]$. Hence,

$$\sum_{i=1}^{J}(i|p) > 0 \qquad \text{(A7.15.3)}$$

Consider now the sum

$$S \triangleq (\text{Sum of quadr. residues}) - (\text{Sum of quadr. nonresidues})$$

$$\Longrightarrow \quad S = \left[\text{Sum of } i=1,2,\ldots,p-1 \ /(i|p)=1\right] - \left[\text{Sum of } i=1,2,\ldots,p-1 \ /(i|p)=-1\right]$$

$$\Longrightarrow \quad S = \left[\text{Sum of } i(i|p) \ /i(i|p)=i,\ i=1,2,\ldots,p-1\right] + \left[\text{Sum of } i(i|p) \ /i(i|p)=-i,\ i=1,2,\ldots,p-1\right]$$

$$\Longrightarrow \quad S = \sum_{i=1}^{p-1} i(i|p) \qquad \text{(A7.15.4)}$$

Using (A7.15.3) it will be shown that, (A7.15.4) is negative, i.e. that the sum of quadratic nonresidues exceeds the sum of quadratic residues (mod p), if $p \equiv 3 \pmod 4$.

$$S = \sum_{i=1}^{p-1} i(i|p) = \sum_{\substack{r=2 \\ r=\text{even}}}^{p-1} r(r|p) + \sum_{\substack{t=1 \\ t=\text{odd}}}^{p-2} t(t|p)$$

Let $r = 2i \ /i=1,2,\ldots,(p-1)/2 = J$ & $t = p-2j$ (=odd) $/j=1,2,\ldots,(p-1)/2 = J$. Then:

$$S = \sum_{i=1}^{J}(2i)(2i|p) + \sum_{j=1}^{J}(p-2j)(p-2j|p)$$

Since $p-2j \equiv -2j \pmod p$, $(p-2j|p) = (-2j|p)$, by (A7.14.1). Then, using Theorem A7.13.2:

$$S = 2(2|p)\sum_{i=1}^{J} i(i|p) + p\sum_{j=1}^{J}(-2j|p) - 2\sum_{j=1}^{J} j(-2j|p) \quad \Longrightarrow$$

$$S = 2(2|p)\sum_{i=1}^{J} i(i|p) + p(-2|p)\sum_{j=1}^{J}(j|p) - 2(-2|p)\sum_{j=1}^{J} j(j|p) \qquad (A)$$

By Theorem (A7.13.2), $(-2|p) = (-1|p)(2|p)$. By Theorem A7.13.1, $(-1|p) \equiv (-1)^{(p-1)/2} \pmod p$. Now, since $(p-1)/2$ = odd by hypothesis, it follows that:

$$\text{If } p \equiv 3 \pmod 4{:} \quad (-1|p) = -1 \qquad (A7.15.5)$$

Hence, $(-2|p) = -(2|p)$. Substituting in eqn (A):

$$S = 2(2|p)\sum_{i=1}^{J} i(i|p) - p(2|p)\sum_{i=1}^{J}(i|p) + 2(2|p)\sum_{i=1}^{J} i(i|p) \longrightarrow$$

$$S = 4(2|p)\sum_{i=1}^{J} i(i|p) - p(2|p)\sum_{i=1}^{J}(i|p) \qquad (B)$$

Following the same technique, S may be expressed differently [from (A7.15.4)]:

$$S = \sum_{i=1}^{J} i(i|p) + \sum_{i=J+1}^{p-1} i(i|p) = \sum_{i=1}^{J} i(i|p) + \sum_{j=1}^{J}(p-j)(p-j|p) \longrightarrow$$

$$S = \sum_{i=1}^{J} i(i|p) + p\sum_{i=1}^{J}(-i|p) - \sum_{i=1}^{J} i(-i|p) \longrightarrow \quad \text{[by (A7.15.5)]}$$

$$S = \sum_{i=1}^{J} i(i|p) - p\sum_{i=1}^{J}(i|p) + \sum_{i=1}^{J} i(i|p) \longrightarrow$$

$$2\sum_{i=1}^{J} i(i|p) = p\sum_{i=1}^{J}(i|p) + S \qquad (C)$$

Substituting (C) in (B):

$$S = 2p(2|p)\sum_{i=1}^{J}(i|p) + 2(2|p)S - p(2|p)\sum_{i=1}^{J}(i|p) \qquad (D)$$

Multiplying both sides of (D), by $(2|p)$ (which is $\neq 0$, because $2 \nmid p$) and noting that $(2|p)^2 = 1$:

$$S(2|p) = p(2|p)^2\sum_{i=1}^{J}(i|p) + 2(2|p)^2 S \longrightarrow$$

$$S(2|p) = p\sum_{i=1}^{J}(i|p) + 2S \quad \longrightarrow$$

$$[2-(2|p)]S = -p\sum_{i=1}^{J}(i|p) < 0 \quad \text{[by (A7.15.3)]} \quad \longrightarrow$$

$$[2-(2|p)]S < 0 \quad \longrightarrow \quad S < 0 \quad [\text{because } 2 > (2|p)]$$

Hence, the sum of quadratic nonresidues exceeds the sum of quadratic residues, so n_E-1 equals the former sum.

Finally, if there was a closed-form expression for n_E, then there would have been one for the sum of quadratic non-residues, and hence for the sum of quadratic residues (mod p) [the two sums add to p(p-1)/2]. This would have solved one of number theory's great problems, because then *Dirichlet's function* L(1) would also have a closed-form expression, for $p \equiv 3 \pmod 4$ (see discussion in H. Davenport [48], pp. 3-11).

QED

APPENDIX 7.16: PROOF OF THEOREM 7.41

Given a (k,J) type-C5 code, since $J \mid k$, one may let $c \triangleq k/J$. Since $n-k = J$, the code length is $n = k+J = cJ+J = (c+1)J$. The code rate is $R \triangleq k/n = (cJ)/[(c+1)J] = c/(c+1)$. Since $m+1 = k$, the actual constraint-length is $n_A \triangleq (m+1)n = kn = cJ(c+1)J = c(c+1)J^2$.

The effective constraint-length, n_E, is bounded by (7.42), for J = odd. The lower bound is $1+(k+1)(J+1)/2 = 1+(k+1)\lceil J/2\rceil$, while the n_E for J = even is $1+(k+1)J/2 = 1+(k+1)\lceil J/2\rceil$. Hence the lower bound on n_E, for J = odd and the exact value of n_E, for J = even, are the same and equal to $1+(k+1)\lceil J/2\rceil = 1+(cJ+1)\lceil J/2\rceil$. The upper bound, from (7.42) and for J = odd, is $1+(cJ+1)(J-1)$.

The ratio $Q \triangleq n_A/n_E$, is bounded, from above, by:

$$c(c+1)J^2/[1+(cJ+1)(J-1)] \le Q \le c(c+1)J^2/[1+(cJ+1)\lceil J/2\rceil]$$

This can be approximated by:

$$c(c+1)J^2/[(cJ+1)(J-1)] \le n_A/n_E \le c(c+1)J^2/[(cJ+1)(J/2)] \Longrightarrow$$

$$c(c+1)J^2/(cJ^2+J-cJ-1) \le n_A/n_E \le 2c(c+1)J/(cJ+1) \Longrightarrow$$

$$c(c+1)J/(cJ+1-c) \le n_A/n_E \le 2c(c+1)J/(cJ) \Longrightarrow$$

$$c(c+1)J/(cJ-c) \le n_A/n_E \le 2(c+1) \Longrightarrow$$

$$(c+1) \le n_A/n_E \le 2(c+1) \Longrightarrow$$

$$[\text{since } R = c/(c+1) \Longrightarrow c = R/(1-R) \Longrightarrow c+1 = 1/(1-R)]$$

$$2(c+1) = 2/(1-R) \le n_A/n_E \le c+1 = 1/(1-R)$$

Obviously, the upper bound is always met for J = even. Note though that the bounds are approximate.

Finally, for J = even,

$$(J/2)/n_E = J/\{2[1+(1+cJ)J/2]\} \approx J/\{2[(1+cJ)J/2] = 1/(1+cJ) \approx$$

$$\approx 1/(cJ) = 1/k$$

QED

APPENDIX 8.1: COMPUTER GENERATION OF 'CYCLIC' CSOC.

This Appendix presents & explains the *flow-charts* of the various number-theoretic routines, used by the simulation programmes. The associated FORTRAN programmes are given in Appendix 8.2.

Note: The expression $a \bmod m$ denotes the least positive residue of a (mod m). This can be obtained from:

$$a \bmod m \;\hat{=}\; a - \lfloor a/m \rfloor m \;=\; a - \mathrm{INT}(a/m)m \qquad (A8.1.1)$$

Also, MAXIN denotes the *maximum integer* of the computer.

A8.1.1. Greatest Common Divisor

This function [IGCD(a,b)]* uses the *Euclidean algorithm* (see Apostol [44], p. 20). See Figure A8.1.1, for a flow-chart.

Specification: "Given integers a & b, return their greatest common divisor, (a,b)". It calls MOD.

```
   If (ab = 0), then: (a,b) = |a+b|  ---------> END
   If (a = b), then: (a,b) = a  --------------> END
   If else, then: A = a & B = b
┌-> Step:   X = A mod B
└─ If (X ≠ 0), then: A = B & B = X & repeat step
   If else, then: (a,b) = B  -----------------> END
```

Figure A8.1.1: Flow-chart for greatest-common divisor.

A8.1.2. Sum Modulo m

This function returns a+b mod m, even if a+b>MAXIN. The algorithm (Fig. A8.1.2) is original. **

Specification: "Given integers a, b & m, return a+b mod m, without causing overflow". It calls MOD, and is based on the following theorem:

Theorem A8.1.1: For any $m \in [1, \mathrm{MAXIN}]$ & any $a,b \in [0,m-1]$,

* See Appendix 8.2 (§ A8.2.1., p. 520).
** This routine is incorporated into other routines.

such that a+b > MAXIN, if MAXr ≙ MAXIN mod m:

$$0 < a+b-MAXIN+MAXr < m \qquad (A8.1.2)$$

Proof: See Appendix 8.2 (§ A8.2.1., p. 520).

∎

A = a mod m & B = b mod m
If (A < MAXIN-B), then: MODSU = A+B mod m ——————————> END
If else, then: MAXr = MAXIN mod m & MODSU = (A-MAXIN)+B+MAXr END

Figure A8.1.2: Flow-chart for a+b mod m.

Note that (A-MAXIN)+B+MAXr ≡ A+B (mod m), because MAXr ≡ MAXIN (mod m). Also, (A-MAXIN)+B+MAXr does not overflow because (A-MAXIN) is evaluated first.

A8.1.3. Product Modulo m

This function [MODPR(a,b,m)]* returns ab mod m, even if ab>MAXIN. The algorithm (Fig. A8.1.3) is original.

Specification: "Given integers a, b & m, return ab mod m, without causing overflow". It calls MOD, ⌊?⌋ ≙ INT(?) & MODSU.

The basic idea behind this routine is that ab = a+a+a+···+a (b-1 additions). To avoid a programme with too many loops (each of which would involve one call to MODSU), it is proposed to add as many as as is possible, without causing overflow, reduce them (mod m) and repeat. To this end, ab is replaced by Qu, Re & MAXr, where Qu is the quotient of the integer division ab/MAXIN, Re the remainder and MAXr ≙ MAXIN mod m [Re ≙ ab - Qu×MAXIN, hence (ab mod m) = (Re mod m) + (Qu mod m)MAXr]. Note that Qu<m because ab < m^2 < m×MAXIN, hence ab/MAXIN<m and ⌊ab/MAXIN⌋ = Qu < m.

What remains to be done is to evaluate Qu×MAXr, without causing overeflow. The same process is followed, i.e. Qu×MAXr = Qu'×MAXIN + Re', where Qu' & Re' are the quotient and the remainder of the division Qu×MAXr/MAXIN; this equals Qu'×MAXr + Re'. Again Qu'×MAXr is expressed as Qu''×MAXr + Re'', until the product between the quotient Qu & MAXr is <

* See Appendix 8.2 (§ A8.2.3., p. 520).

MAXIN. Then, MODPR = Re+Re'+Re''+···+QuxMAXr. That this expression will not cause overflow, is decided by QuxMAXr < MAXIN <——> Qu < MAXIN/MAXr <——> Qu < ⌊MAXIN/MAXr⌋.

```
       A = a mod m  &  B = b mod m
   If (A = 0, or B = 0, or A ≤ MAXIN/B), then:  MODPR = AB mod m     END
   If else, then: D = (A/MAXIN)B, MAXr = MAXIN mod m
                      Qu = INT(D) & Re = D-QuxMAXIN mod m
   If (MAXr = 0), then:  MODPR = Re  ————————————————————————>  END
   If else, then:  Dx = MAXr/MAXIN  &  LIM = INT(MAXIN/MAXr)
┌─>  Step:    If (Qu ≤ LIM), then: MODPR = MODSU(QuxMAXr,Re,m)  ——>  END
│             If else, then: D = DxxQu & Qu = INT(D),
│                                 Rs = D-QuxMAXIN mod m & Re = MODSU(Re,Rs,m)
└──────────   Repeat step
```

Figure A8.1.3: Flow-chart for ab mod m.

A8.1.4. Power Modulo m

This function [MODRE(a,b,m)]* returns a^b mod m, even if a^b>MAXIN. The algorithm (Fig. A8.1.4) is original.

Specification: "Given integers a, b & m, return a^b mod m, without causing overflow". It calls MOD, ⌊?⌋ ≙ INT(?), MODSU & MODPR.

```
       MAXg = logMAXIN  &  A = a mod m  &  B = b  &  C = 1
┌─>  Step:    If (A ≤ 1), then:  MODRE = A  ———————————————>        END
│             If else, then:  k = INT(MAXg/logA)
│  If (B < k), then:   z = A^B mod m  &  MODRE = MODPR(z,C,m)     END
│  If else, then:
│  If (k = 1), then:   If (B = odd), then: C = MODPR(C,A,m)
│                      A = MODPR(A,A,m)  &   B = INT(B/2)
├────────────────────  Repeat step
│  If (k ≠ 1), then:   R = B mod k  &  Cn = A^R mod m  &  B = INT(B/k)
│                      C = MODPR(C,Cn,m)  &  A = A^k mod m
└────────────────────  Repeat step
```

Figure A8.1.4: Flow-chart for a^b mod m.

Since $a^b = a^{qk+r}$, where q = ⌊b/k⌋, and r = b-qk, a^b may be written as $a^b = (a^k)^q a^r$, where k is chosen so that a^k < MAXIN <——> k < logMAXIN/loga. A = a^k mod m is the new base, B = q is the new exponent and C = a^r (mod m) multiplies the final result. This is repeated until A^B < MAXIN. If, and when, A^2 > MAXIN the new values of A, B & C are C = MODPR(A,C,m) if B = odd, A = MODPR(A,A,m) and B = ⌊B/2⌋.

* See Appendix 8.2 (§ A8.2.4., p. 521).

A8.1.5. Prime Decomposition

This subroutine [PRIDE1(m,Arr)]* returns the prime decomposition of m. The algorithm (Fig. A8.1.5) is original.

Specification: "Given integer m, return its prime factors $p_1<p_2<\cdots<p_r$, and their respective exponents $a_1,a_2,\ldots,a_r$, in 22 x 2 array Arr [so that Arr(i,1) = p_i & Arr(i,2) = a_i]. The rest of the array should be zero." It calls MOD, $\lfloor ? \rfloor \hat{=}$ INT(?) & SQRT.

This subroutine generates test integers p = 2,3,5,7,9,.., starting with p_1 = 2. It examines whether $p_1 \mid m$; if it does, m is reduced to m/p_1, and a_1 is increased by 1, and repeats until p_1 does not divide m. It then updates the array (if $a_1 \geq 1$), so that Arr(1,1) = p_1 & Arr(1,2) = a_1, and considers the next integer p_2, unless m = 1 (in which case it terminates). If $p_2 = 3 \nmid m$, it considers p_3 = 5, etc until it obtains another divisor of m. In this way only prime integers are considered, because if, say, 9 is tested it will not divide m since 3 has already been tested and all

```
      Arr = 0  &  M = m  &  p = 0  &  i = 0
--> Step 1:  a = 0  &  p = p+2  &  If (p=4), then:  p = 3
 --> Step 2:  If (M mod p = 0), then:  M = M/p  &  a = a+1
                                       Repeat step 2
    If else, then:  If (a = 0), then:  Go to step 3
                    If (a ≠ 0), then:  i = i+1  &  Arr(i,1) = p
                                             Arr(i,2) = a
     Step 3:  If (p ≤ √M), then:  Repeat step 1  <--
              If else, then:  If (M = 1), then:  ---------->  END
                              If else, then:  Arr(i+1,1) = M
                                              Arr(i+1,2) = 1  END
```

Figure A8.1.5: Flow-chart for prime decomposition of m.

factors equal to 3 have been removed via the m = m/p operation. The search terminates if m = 1, but processing may be sped up if one considers the fact that if m is not divided by any of $2,3,\ldots,\sqrt{m}$, then it is only divided by itself. Hence, the test is repeated until $p > \sqrt{m}$. Then m is a prime and the search terminates there.

Subroutine PRIDE2* does what PRIDE1 does (by calling it) and it also returns r (the number of prime factors of m) and

* See Appendix 8.2 (§ A8.2.5., p. 522).

$\theta(m)$. Note also that an array with up to 22 prime factors can accomodate the prime decomposition of integers of the order of 3×10^{30} (= the product of the 22 smallest primes).

A8.1.6. Primitive Root Modulo m

This function [IPRIM1(m)]* returns the smallest primitive root of m. The algorithm (Fig. A8.1.6) is original.

Specification: "Given integer $m\in[1,MAXIN]$, return the smallest primitive root (mod m). If m has no primitive root, return 0". It calls MOD, MODRE & PRIDE1.

The straightforward approach is to test integers n = 1,2,...,m-1 /(n,m) = 1, until a primitive root is found. The test is $n^i \not\equiv 1$ (mod m), for $i=1,2,\ldots,\Phi(m)-1$. It is obvious that such an approach is inefficient.

Firstly, the special cases are examined. For m = 1,2,3 & 4, g = m-1 /m>1 and g = 1 /m=1 (see Apostol [44], p. 205). Next, the prime decomposition of m is obtained. For m>4, m has a prim. root only if $m = p^a$, or $m = 2p^a$ /a≥1 (ibid). Then, if r>2, or if r=2 & $p_1>2$, or if r=2 & $p_1=2$ & $a_1>1$, there is no prim. root and g = 0. Finally, if $p_1=3$ then $m = 3^a$ /a≥2, or $m = 2 \times 3^a$ /a≥1. For the former case, g=2 is a prim. root (mod 3). Since $2^{p-1} = 2^2 = 4 \equiv 4 \pmod{3^2}$, then (by Theorem A7.3.4) g=2 is also a prim. root (mod 3^a). For the latter case, g=5 is also a prim. root (mod 3^a) because 5 [$5 \equiv 2$ (mod 3)] is a prim. root (mod 3) and because $5^{p-1} = 25 \equiv 7$ (mod 9) is also a prim. root (mod 3^a) (by Theorem A7.3.4). Since 5 is odd, then it is a prim. root (mod $2p^a$) (ibid, p. 210). Furthermore, 5 is the smallest prim. root because 2,3 & 4 are not relatively prime to the modulus. Hence, if $p_1=3$, then g=3r-1.

For the remaining of the cases ($m=p^a$, or $m=2p^a$ /p≥5), what is required is the smallest prim. root (mod p), or the smallest odd prim. root (mod p) (if $2|m$). In both cases, g should satisfy $g^{p-1} \not\equiv 1 \pmod{p^2}$, if a>1.

To this end candidate gs are generated and tested. If $2|m$, then g=3,5,7,..., otherwise g=2,3,4,... The search ends if $g\geq p-1=\Phi(p)=\Phi$ (p-1 has order 2, while g has order p-1≥4). If this happens (it should not) then p is replaced by p^a and

* See Appendix 8.2 (§ A8.2.6., p. 524).

the search starts again for $g=p+1,p+2,\ldots$ (if $2\nmid m$), or $g=p+2,p+4,\ldots$ (if $2|m$). If it fails again (it should not) then $g=-1$ and the search terminates.

Returning now to the normal search, every g is tested to determine if $g^i \not\equiv 1 \pmod p$, for $i<p-1=\Phi$. If the test fails, another g is considered [note that if p^a is tested, instead of p, then $i<\Phi(p^a)=p^{a-1}(p-1)=\Phi$]. To speed-up the process, not all is are considered, but only those that may result in $g^i \equiv 1 \pmod p$. According to Theorem A7.3.2, if $i\nmid p-1$ then $g^i \not\equiv 1 \pmod p$. Furthermore, if $q_1,q_2,\ldots,q_s$ are the prime factors of Φ and $i=\Phi/q_j$ satisfies the test $g^i \not\equiv 1 \pmod p$, g also satisfies the test for other powers of q_j (provided that they divide Φ), because: If $g^i \not\equiv 1 \pmod p$ and $g^{[\Phi/(q_j q_x)]} \equiv 1 \pmod p$, then $g^{[\Phi/(q_j q_x)]q_x} = g^{[\Phi/(q_j)]} = g^i \equiv 1 \pmod p$, which contradicts the hypothesis. Hence, it is enough to test all exponents Φ/q_j /$j=1,2,\ldots,s$.

```
If (m = 1), then:  g = 1  ——————————————————————————> END
If (2≤ m ≤ 4), then:  g = m-1  —————————————————————> END
If (m ≥ 5, & m is not p^a, or 2p^a), then:  g = 0  ———> END
If else, let r=1 if m=p^a  or  r=2 if m=2p^a
If (p=3), then:  g = 3r-1  —————————————————————————> END
If else, then:  Obtain the prime factors (q_1,q_2,...,q_s) of Φ = p-1
                g = 1
┌─> Step:  g = g+r
│          If (g ≥ p-1), then:  Repeat for p = p^a  &  Φ = p^(a-1)(p-1)
│                               If it fails:  g = -1  &  STOP
│          If (g < p-1), then:
├── If {g^(Φ/q_j) ≡ 1 (mod p), for at least one j∈[1,s]}, then:  Repeat step
│   If else, then:  If (a = 1), then:  ———————————————> END
│                   If (a > 1), then:
│                               If [g^(p-1) ≢ 1 (mod p²)], then: END
└───────────────────────────    If else, then:  Repeat step.
```

Figure A8.1.6: Flow-chart for the smallest primitive root.

A8.1.7. Order J Modulo any Divisor d>1 of m

This function [IEXP1(m,J)]* returns an element β ($1\le\beta\le m$) of order J modulo any divisor $d>1$ of m. The algorithm (Fig. A8.1.7) is original.

Specification: "Given integers m & J, return element $\beta\in[1,m]$, such that $\mathrm{Ord}_d(\beta) = J$, for all $d|m$ & $d>1$. If $m<3$, or $J<2$, or m=even & $J\neq 2$, or m=odd & $J\nmid\theta(m)$, return 0". It

* See Appendix 8.2 (§ A8.2.7., p. 525).

calls MOD, PRIDE2, IPRIM1, MODRE, MODPR & MODSU.

The routine is a straightforward application of eqn (7.33a). It also includes checks for illegal pairs of m & J.

If ($m \le 2$ or $J \le 1$), *then*: $\beta = 0$ ⟶ END
If ($J = 2$), *then*: $\beta = m-1$ ⟶ END
If ($J > 2$ & m = odd), *then*: $\beta = 0$ ⟶ END
If else, *then*: Obtain prime decomposition of m
($p_1, p_2, \ldots, p_r$ & $a_1, a_2, \ldots, a_r$) and $\theta(m)$
If [$J \nmid \theta(m)$], *then*: ⟶ END
If else, *then*: Let $\beta = 0$
For $i = 1, 2, \ldots, r$
$m_i = p_i^{a_i}$ & $f_i = (m_i/p_i)(p_i - 1)$ & $g_i = \mathrm{IPRIM1}(m_i)$
$A = \mathrm{MODRE}(g_i, f_i/J, m)$ & $B = \mathrm{MODRE}(m/m_i, f_i, m)$
$C = \mathrm{MODPR}(A, B, m)$ & $\beta = \mathrm{MODSU}(\beta, C, m)$
END

Figure A8.1.7: Flow-chart for the IA generator-element ß.

A8.1.8. Encoding and Syndrome Arrays

This routine [CODAR2(k+1,J,Jarr,Karr)]* returns the encoding & the syndrome array, for the (k,J) type-C5, or the k type-B4 code. The algorithm (Fig. A8.1.8) is original.

Specification: "Given integers k & J, return the EA in array Jarr & the SA in array Karr, for the (k,J) type-C5 code, or the k type-B4 code. If there is no code, return Jarr(2,1) < 0". It calls MOD & IEXP1.

The work starts with the calculation of ß. If IEXP1 = 0, the routine terminates, returning Jarr(2,1) < 0.

The EA is obtained from the IA via mapping (7.35). To save space, the IA is not calculated but array Karr is used as a working array for the storage of the coset leaders of the IA. The first column of the IA is stored in Karr(i,1) $\equiv \beta^i$ (mod k+1) /i=1,2,...,J. The 1st row of Jarr is set equal to 0. Jarr is calculated coset by coset, starting from coset number, C_n = 1. Hence, scanning Jarr(1,i), as i=1,2,...,k, permits the determination of the coset leader, C_r, of the next coset to be calculated [Jarr(1,i)=0, for the smallest i]. Once a coset leader, C_r, is found, the corresponding column of Jarr is obtained from the mapping:

$$\mathrm{Jarr}(i, C_r) = C_n + (i-1)k/J \quad /i=1,2,\ldots,J \qquad (A8.1.3)$$

* See Appendix 8.2 (§ A8.2.8., p. 525).

The rest of the columns that belong to coset C_n, are the rest of the elements of column C_r, of the IA (C_r is the last element of that column). By Definition 7.2 (p. 185), the IA elements of column z are $a_{x,z} \equiv za_{x,1}$ (mod k+1) /x=1,2,...,J. Since Karr(i,1) contains the IA first-column elements, then the IA elements of column C_r are obtained by:

$$Cl_n(i) = C_r \times Karr(i,1) \bmod k+1 \quad /i=1,2,...,J \qquad (A8.1.4)$$

$Cl_n(i)$ /i=1,2,...,J are both the IA elements of the coset leader of coset C_n [which were mapped to C_n+(i-1)k/J - by (A8.1.3)] and also the column numbers of coset C_n. According to (7.37a) (p. 221), the column numbers of an IA coincide with the elements of the Jth row, $Cl_n(i)$ /i=1,2,...,J and also the last elements of the IA columns belonging to coset C_n. Hence, the last elements of these columns of the IA are also mapped to C_n+(i-1)k/J:

$$Jarr(J,Cl_n(i)) = C_n + (i-1)k/J \quad /i=1,2,...,J-1 \qquad (A8.1.5)$$

Note, from (A8.1.4), that $Cl_n(J) = C_r$, and that Jarr(J,C_r) has been calculated. The rest of the elements of column $Cl_n(i)$ are calculated using the cyclic nature of the EA. So, the element in row J-1 will be smaller by k/J, unless this is non-positive, in which case k must be added:

$$Jarr(j,Cl_n(i)) = Jarr(j+1,Cl_n(i))-k/J^* \quad /j=J-1,..,2,1 \quad (A8.1.6)$$

The search terminates when C_n > k/J.

Consider now the *syndrome array* (SA). Let the syndrome register (see Fig. 7.1) be rotated 90° clockwise and then 180° around its vertical axis, to become the k x J array ISR. Then, the top row contains $s^{(j)}_{h+k-1}$ /j=1,2,...,J, the 2nd row $s^{(j)}_{h+k-2}$ /j=1,2,...,J, etc, the last row contains $s^{(j)}_{h}$ /j=1,2, ..,J. Obviously, if h is the block currently decoded:

$$ISR(z,j) = s^{(j)}_{h+k-z} \quad /j=1,2,...,J \ \& \ z=1,2,...,k \qquad (A8.1.7)$$

Theorem 7.1 relates the syndrome bits with the elements of the IA. Since decoding is done via the EA, the latter's elements are used instead. By Theorem 7.1, for each i=1,2,.. .,k, $s^{(j)}_{h+w-1}$ checks on $e^{(i)}_h$, iff EA element $b_{j,w}$ = i. Since, $b_{j,w}$

* Add k, if the RHS is less than one.

= Jarr(j,w) and since there is exactly one i in each EA row, then for each j=1,2,...,J, there exists a column z (obviously $1 \le z \le k$), such that Jarr(j,z) = i. Then, $e_h^{(i)}$ is checked by $s_{h+z-1}^{(j)}$ = ISR(k+1-z,j). For each i = 1,2,...,k, syndrome bits ISR(k+1-z,j) /j=1,2,..,J, check on $e_h^{(i)}$, where z is such that Jarr(j,z) = i. Obviously, z depends on j & i. Hence, an expression for z(j,i) is required and since z is a column of Jarr, $z \in [1,k]$. This expression is obtained from the inversion of Jarr(j,z(j,i)) = i. Let z(j,i) = x /x=1,2,...,k. Then, Jarr(j,x) = i and from z(j,i) = x, one obtains z(j,Jarr(j,x)) = x /x=1,2,...,k. To facilitate decoding, let Karr(j,i) = k+1-z(j,i). Then:

Karr(j,Jarr(j,i)) = k+1-i   /i=1,2,...,k & $1 \le j \le J$ (A8.1.8)

Then, ISR(Karr(j,i),j) /j=1,2,...,J are the syndrome bits checking on $e_h^{(i)}$, for each i = 1,2,...,k. See Example A8.2.1 (p. 531), for an illustration of the validity of the above results, via the calculation of the EA & SA for the (18,6) type-C5 code.

In Appendix 8.2 (§ A8.2.8., pp. 525-31), FORTRAN programmes for subroutines CODAR1 & CODAR3 are also listed. The first one prints any combination of the IA, EA & SA, required. Each array is partitioned into sub-arrays of dimensions that fit in the printer paper. CODAR3 returns the same

ß = Karr(1,1) = IEXP1(k+1,J)
<u>If</u> (ß = 0), <u>then</u>: Jarr(2,1) < 0 $\longrightarrow$ <u>END</u>
<u>If</u> else, <u>then</u>:
For i=2,3,...,J
Karr(i,1) = ßxKarr(i-1,1) mod k+1
Jarr(1,i) = 0 /i=1,2,...,k & $C_r = C_n = 0$
<u>Step</u>: $C_r = C_r + 1$
<u>If</u> [Jarr(1,C_r) $\neq$ 0], <u>then</u>: <u>Repeat</u> step
<u>If</u> else, <u>then</u>: $C_n = C_n+1$ & Jarr(i,C_r) = C_n + (i-1)k/J /i=1,2,...,J
For i=1,2,...,J-1
Cl_n(i) = C_rxKarr(i,1) mod k+1
Jarr(J,Cl_n(i)) = C_n + (i-1)k/J
For j=J-1,...,2,1
Jarr(j,Cl_n(i)) = Jarr(j+1,Cl_n(i)) - k/J
<u>If</u> [Jarr(j,Cl_n(i)) < 1] <u>then</u>: Increase by k
<u>If</u> (C_n < k/J), <u>then</u>: <u>Repeat</u> step
Karr(j,Jarr(j,i)) = k+1-i /i=1,2,...,k & j=1,2,...,J <u>END</u>

Figure A8.1.8: Flow-chart for the encoding & syndrome arrays.

information, as CODAR2, but in a way suitable for the decoder implementation of 'long' codes (see Examples A8.3.1 & A8.3.2 in Appendix 8.3, pp. 536 & 541).

A8.1.9. Effective Constraint-Length

This function [NEFEL1(k+1,J)]* returns the effective constraint-length, n_E, of the (k,J) type-C5, or the k type-B4 code. The algorithm (Fig. A8.1.9) is original.

Specification: "Given integers k+1 & J, return the effective constraint-length of the (k,J) type-C5, or the k type-B4 code. If no such code exists, return 0". It calls MOD, MAX & IEXP1.

This routine calls IEXP1 to calculate ß. If ß=0, it returns 0. If J=even, it returns 1+(k+1)J/2**, while if J=3, it returns 1+2(k+1)**. For the rest of the cases, it examines the IA to determine the column with the largest sum of elements. To avoid having to store the IA, it generates only the coset leaders, from ß. An 1 x k logical array, Isu, is used to 'tick-off' examined IA columns. To speed-up the process, if a column sums-up to the maximum [(k+1)(J-1), by Theorem 7.39] the search terminates. Also, if the opposite happens (by Theorem 7.35, the minimum is k+1), another column will sum-up to the maximum.

```
      ß = IEXP1(k+1,J)  &  n_E = 0
If (ß = 0), then: ------------------------------------------> END
If (J = even), then:  n_E = 1+(k+1)J/2 ---------------------> END
If (J = 3), then:  n_E = 1+2(k+1) --------------------------> END
If else, then:
            Isu(i) = F  /i=1,2,...,k  &  C_r = C_n = 0
     -----> Step:  C_r = C_r + 1
            If [Isu(C_r) = T], then:  Repeat step
            If else, then:  C_n = C_n + 1  &  C_nn = C_r  &  Sum = 1
     -----> For j=1,2,...,J
            C_nn = ßxC_nn mod k+1   &   Isu(C_nn) = T
            Sum = Sum + C_nn  &  n_E = MAX{n_E,Sum}
            If (Sum = k+2), then:  n_E = 1+(k+1)(J-1)
            If [n_E = 1+(k+1)(J-1)], then: ---------------> END
            If (C_n < k/J), then:  Repeat step
            If else, then: -----------------------------> END
```

Figure A8.1.9: Flow-chart for the effective constraint-length.

* See Appendix 8.2 (§ A8.2.9., p. 533).
** See Theorems 7.38 & 7.39 (p. 224).

APPENDIX 8.2: FORTRAN PROGRAMMES FOR APPENDIX 8.1

A8.2.1. Greatest Common Divisor

```
      FUNCTION IGCD(K,M)
C  IGCD = G.C.D.(K,M).-
      I1=K
      I2=M
  140 I3=MOD(I1,I2)
      IGCD=I2
      IF(I3.EQ.0) RETURN
      I1=I2
      I2=I3
      GO TO 140
      END
```

Figure A8.2.1 FORTRAN programme for *function IGCD*.

A8.2.2. Sum Modulo m

To prove Theorem A8.1.1: Since,

$0 \le a,b < m \Longrightarrow MAXIN < a+b < 2m \Longrightarrow$

$0 < a+b-MAXIN < 2m-MAXIN \Longrightarrow$

$0 < a+b-MAXIN+MAXr < 2m-MAXIN+MAXIN-\lfloor MAXIN/m \rfloor m \Longrightarrow$

$$0 < a+b-MAXIN+MAXr < m+(1-\lfloor MAXIN/m \rfloor)m \qquad (A)$$

Since $m \le MAXIN \Longrightarrow \lfloor MAXIN/m \rfloor \ge 1 \Longrightarrow$

$1-\lfloor MAXIN/m \rfloor \le 0$. From this & (A), (A8.1.2) follows.

QED

A8.2.3. Product Modulo m

This subroutine was tested for validity for various moduli up to MAXIN-1. A processing-time test was carried out for various moduli of magnitude-order 10^3-10^7 on an ICL-1904 mainframe (for which MAXIN = 8,388,607). For each modulus, the routine was called about 5,000 times. The 1st factor of the product was ≈ m, while the 2nd factor ≈ m/2. The average processing time was 11, 12, 15, 27 & 16 μsec, respectively, for the above-mentioned moduli.

```
      FUNCTION MODPR(IA,IB,IMO)
C  MODPR = IA*IB (MODULO IMO).-
      DOUBLE PRECISION D,DCM
      COMMON/MO/ICE
      M1=MOD(IA,IMO)
      M2=MOD(IB,IMO)
      IF(M1.NE.0.AND.M2.NE.0) GO TO 190
  170 MODPR=0
      RETURN
  190 IF(M1.GT.ICE/M2) GO TO 220
      MODPR=MOD(M1*M2,IMO)
      RETURN
  220 D=DBLE(FLOAT(M1))/ICE*M2
      M2=MOD(IDINT(D),IMO)
      IR=MOD(IDINT((D-M2)*ICE+0.5),IMO)
      ICM=MOD(ICE,IMO)
      DCM=DBLE(FLOAT(ICM))/ICE
      IF(ICM.EQ.0) GO TO 170
      LIM=ICE/ICM
  290 IF(M2.LE.LIM) GO TO 390
      D=DCM*M2
      M2=MOD(IDINT(D),IMO)
      IRR=MOD(IDINT((D-M2)*ICE+0.5),IMO)
  330 IF(IR.LE.ICE-IRR) GO TO 370
      IR=MOD(IR-ICE+IRR,IMO)
      IRR=ICM
      GO TO 330
  370 IR=MOD(IR+IRR,IMO)
      GO TO 290
  390 MODPR=MOD(M2*ICM,IMO)
  400 IF(MCDPR.LE.ICE-IR) GO TO 440
      MODPR=MOD(MODPR-ICE+IR,IMO)
      IR=ICM
      GO TO 400
  440 MODPR=MOD(MODPR+IR,IMO)
      RETURN
      END
```

Figure A8.2.2: FORTRAN programme for *function MODPR.*

A8.2.4. Power Modulo m

This subroutine was tested for validity and processing-time performance, for various moduli up to MAXIN-1, in a way similar to that used for MODPR. Again, the moduli order of magnitude was 10^3-10^7 (on an ICL-1904 mainframe, MAXIN = 8,388,607). For each modulus, the routine was called about 5,000 times. The base was between 300 & 350 and the exponent between 151 & 170. The average processing-time was 12, 31, 24, 72 & 47 µsec, respectively, for the above-mentioned moduli.

```
      FUNCTION MODRE(IBA,IEX,IMO)
C  MODRE = IBA**IEX (MODULO IMO).-
      COMMON/MO/ICE
      A=ALOG(FLOAT(ICE))
      IB=MOD(IBA,IMO)
      IE=IEX
      IC=1
  190 IF(IB.GT.1) GO TO 230
      MODRE=0
      IF(IB.EQ.1) MODRE=IC
      RETURN
  230 K=A/ALOG(FLOAT(IB))
      IF(IE.GE.K) GO TO 280
      MODRE=MOD(IB**IE,IMO)
      MODRE=MODPR(MODRE,IC,IMO)
      RETURN
  280 IF(K.EQ.1) GO TO 330
      IC=MODPR(IC,MOD(IB**MOD(IE,K),IMO),IMO)
      IB=MOD(IB**K,IMO)
      IE=IE/K
      GO TO 190
  330 IF(MOD(IE,2).EQ.1) IC=MODPR(IC,IB,IMO)
      IB=MODPR(IB,IB,IMO)
      IE=IE/2
      GO TO 190
      END
```

Figure A8.2.3: FORTRAN programme for *function MODRE.*

A8.2.5. Prime Decomposition

This subroutine was tested for validity and processing-time performance for various moduli up to MAXIN. For the ICL-1904 mainframe (with MAXIN = 8,388,607), it was verified that: 8,388,607 = 47 x 178,481

8,388,606 = 2 x 3 x 23 x 89 x 683

8,388,605 = 5 x 1,677,721

8,388,604 = 2^2 x 7^2 x 127 x 337

8,388,593 = prime

etc

```
      SUBROUTINE PRIDE1(NI,IAR)
C  THIS SUBROUTINE RETURNS THE PRIME DECOMPOSITION OF NI, IN 22X2 ARRAY
C IAR - PRIMES IN 1ST COLUMN, CORRESPONDING EXPONENTS IN 2ND - THE PRI-
C MES AND THEIR EXPONENTS ARE ARRANGED IN ASCENDING ORDER AND OCCUPY THE
C FIRST ROWS OF THE ARRAY, WHILE THE REST ROWS ARE Ø.-
C  1<IMO<MAXINT+1 / MAXINT<2**1Ø1 - IF IMO<2, IAR=Ø - NI IS RETURNED.-
      DIMENSION IAR(22,2)
      DO 19Ø I=1,22
      IAR(I,1)=Ø
  19Ø IAR(I,2)=Ø
      IF(NI.LT.2) RETURN
      NNI=NI
      J=Ø
      I=Ø
      LIM=(SQRT(FLOAT(NI))+1)/2
  25Ø I=I+1
      IPR=2*I-1
      IF(I.EQ.1) IPR=2
      IEX=Ø
  29Ø IF(MOD(NNI,IPR).NE.Ø) GO TO 33Ø
      NNI=NNI/IPR
      IEX=IEX+1
      GO TO 29Ø
  33Ø IF(IEX.EQ.Ø) GO TO 38Ø
      J=J+1
      IAR(J,1)=IPR
      IAR(J,2)=IEX
      IF(NNI.EQ.1) RETURN
  38Ø IF(I.LE.LIM) GO TO 25Ø
      IF(NNI.EQ.Ø) RETURN
      IAR(J+1,1)=NNI
      IAR(J+1,2)=1
      RETURN
      END

      SUBROUTINE PRIDE2(IMO,IL,NR,KAR)
C  THIS SUBROUTINE RETURNS THE PRIME DECOMPOSITION OF IMO IN 22X2 ARRAY
C KAR - PRIMES IN 1ST COLUMN, CORRESPONDING EXPONENTS IN 2ND - THE PRI-
C MES AND THEIR EXPONENTS ARE ARRANGED IN ASCENDING ORDER AND OCCUPY THE
C FIRST NR ROWS OF THE ARRAY, WHILE THE REST ROWS ARE Ø.-
C  THE SUBROUTINE CALCULATES AND RETURNS IL AND NR, WHERE NR IS THE NUM-
C BER OF PRIME DIVISORS, IPR(1),IPR(2),...,IPR(NR), OF IMO, AND
C IL = G.C.D.( IPR(1)-1,IPR(2)-1,...,IPR(NR)-1 ).-
C  1<IMO<MAXINT+1 / MAXINT<2**1Ø1 - IF IMO<2, KAR=Ø, NR=Ø, IL=-1.-
      DIMENSION KAR(22,2)
      CALL PRIDE1(IMO,KAR)
      NR=Ø
      IL=KAR(1,1)-1
      IF(IL.EQ.-1) RETURN
      DO 26Ø I=1,22
      IF(KAR(I,1).EQ.Ø) GO TO 27Ø
  26Ø NR=NR+1
  27Ø IF(NR.EQ.1) RETURN
      DO 29Ø I=2,NR
  29Ø IL=IGCD(IL,KAR(I,1)-1)
      RETURN
      END
```

Figure A8.2.4: FORTRAN programmes for *subroutines PRIDE?.*

A8.2.6. Primitive Root Modulo m

This subroutine was tested for validity and processing-

```
      FUNCTION IPRIM1(IMO)
C  IPRIM1=SMALLEST PRIMITIVE ROOT OF IMO.-
C  IF IMO HAS NO PRIMITIVE ROOTS, IPRIM1=0.-
C  IF NO PRIMITIVE ROOT IS FOUND, IPRIM1=-1.-
      INTEGER X02BBF
      DIMENSION IAR(10,2),JAR(10,2)
      COMMON/MO/ICE
      ICE=X02BBF(X)
      IPRIM1=1
      IF(IMO.GE.5) GO TO 190
      IF(IMO.GT.2) IPRIM1=IMO-1
      RETURN
  190 IPRIM1=0
      IND=1
      CALL PRIDE1(IMO,IAR)
      IF(IAR(1,1).GE.3.AND.IAR(2,1).EQ.0) GO TO 250
      IND=2
      IF(IAR(1,1).EQ.2.AND.IAR(1,2).EQ.1.AND.IAR(2,1).NE.0.AND.IAR(3,1).
     1EQ.0) GO TO 250
      RETURN
  250 IPR=IAR(IND,1)
      IF(IPR.NE.3) GO TO 290
      IPRIM1=3*IND-1
      RETURN
  290 IA=IAR(IND,2)
      LIM=IPR-1
      CALL PRIDE1(LIM,JAR)
      I=1
  330 I=I+IND
      IF(I.LE.IPR-2) GO TO 440
      IF(IA.NE.0) GO TO 380
      IPRIM1=-1
      RETURN
  380 I=IPR+IND
      LIM=(IPR-1)*IPR**(IA-1)
      IPR=IPR**IA
      CALL PRIDE1(LIM,JAR)
      IA=0
      GO TO 330
  440 J=1
  450 IF(MODRE(I,LIM/JAR(J,1),IPR).EQ.1) GO TO 330
      J=J+1
      IF(JAR(J,1).NE.0) GO TO 450
      IF(IA.EQ.0.OR.IAR(IND,2).EQ.1) GO TO 500
      IF(MODRE(I,IPR-1,IPR**2).EQ.1) GO TO 330
  500 IPRIM1=I
      RETURN
C  INCORPORATE FUNCTIONS MODPR & MODRE  AND SUBROUTINE PRIDE1.-
      END
```

Figure A8.2.5: FORTRAN programme for *function IPRIM1.*

time performance for various moduli up to MAXIN. On an ICL-1904, it required 33 secs to examine m = 1-1000, 30 secs for 1001-2000, 35 secs for 2001-3000, 58 secs for 5001-6000, 99 secs for 8001-9000 and 139 secs for 8388400-8388607 (MAXIN = 8,388,607). For m = 8,388,602, g = 7.

A8.2.7. Order J Modulo any Divisor d>1 of m

This subroutine was tested for validity and processing-speed performance for various moduli. For a given m between 2000-3000, it required about 0.4 secs to calculate ß, where $Ord_d\beta$ = J, for all J | θ(m) (on a CDC-7600 mainframe). Note that IEXP1 calls PRIDE2, IPRIM1, MODRE & MODPR.

```
      FUNCTION IEXP1(M,IS)
      DIMENSION KAR(22,2),IPEX(22),IREX(22),IPR(22)
      COMMON/TP1/IL,NR,KAP
      IF(M.GT.2.AND.IS.GT.1.AND.(MOD(M,2).EQ.1.OR.IS.EQ.2)) GO TO 290
  270 IEXP1=0
      RETURN
  290 IEXP1=M-1
      IF(IS.EQ.2) RETURN
      CALL PRIDE2(M,IL,NR,KAP)
      IF(MOD(IL,IS).NE.0) GO TO 270
      IEXP1=0
      DO 400 I=1,NR
      IPEX(I)=KAR(I,1)**KAR(I,2)
      IREX(I)=IPEX(I)/KAR(I,1)*(KAR(I,1)-1)
      IPR(I)=IPRIM1(IPEX(I))
      IA=MODRE(IPR(I),IREX(I)/IS,M)
      IB=MODRE(M/IPEX(I),IREX(I),M)
  400 IEXP1=MOD(IEXP1+MODPR(IA,IB,M),M)
      RETURN
      END
```

Figure A8.2.6: FORTRAN programme for *function IEXP1.*

A8.2.8. Encoding and Syndrome Arrays

Subroutine CODAR2 returns the encoding (EA) & syndrome (SA) arrays in the J x k arrays JAR & KAR. It is used by the simulation programmes, for decoding.

Subroutine CODAR1 prints any combination of the IA, EA & SA, without making use of storage arrays. The arrays are partitioned so that they can fit in the available printer paper (see Fig. A8.2.8).

```
      SUBROUTINE CODAR2(K0,IS,JAR,KAR)
      DIMENSION JAR(IS,K0),KAR(IS,K0)
      COMMON/COD2/JEXP
      M=K0+1
      JEXP=KAR(1,1)=IEXP1(M,IS)
      IF(KAR(1,1).EQ.0) RETURN
      IA=JAR(2,1)
      IB=JAR(1,1)
      DO 210 I=2,IS
  210 KAR(I,1)=MOD(KAR(I-1,1)*KAR(1,1),M)
      DO 230 I=1,K0
  230 JAR(1,I)=0
      NCS=(-1)*IS
      N=0
  260 N=N+1
      IF(JAR(1,N).NE.0) GO TO 260
      NCS=NCS+IS
      NCC=NCS+IS+1
      DO 310 I=1,IS
  310 JAR(I,N)=NCS+I
      DO 370 I=2,IS
      K=MOD(N*KAR(I-1,1),M)
      JAR(1,K)=NCS+I
      DO 370 J=2,IS
      JAR(J,K)=JAR(J-1,K)+1
  370 IF(JAR(J,K).EQ.NCC) JAR(J,K)=NCS+1
      IF(NCC.LT.M) GO TO 260
      IC=IA*(IB-1)
      DO 410 I=1,IS
      DO 410 J=1,K0
  410 KAR(I,JAR(I,J))=J*IA+IC
      RETURN
      END
```

Figure A8.2.7: FORTRAN programme for *subroutine CODAR2.*

```
      SUBROUTINE CODAR1(M,IS,ID)
      DIMENSION IR(200),IFC(200),IFR(200)
C  INCORPORATE THE FOLLOWING STATEMENT, IN THE CALLING SEGMENT.-
      COMMON/COD1/IA,IB/COD2/JEXP
      DATA K3/L/,E/'-'/,H/'+'/,F/':'/,G/' '/
      IFC(1)=JEXP
      IF(IFC(1).NE.0) GO TO 315
      ID=0
  310 RETURN
  315 IF(ID.LT.1.OR.ID.GT.7) ID=1
      NCP=39-INT(ALOG10(M-1.0))/2*10
      IDD=0
      K0=M-1
      NPW=K0/NCP
      NRC=K0-NPW*NCP
      NRP=50
      NPL=IS/NRP
      NRR=IS-NPL*NRP
      DO 365 I=2,IS
  365 IFC(I)=MOD(IFC(I-1)*IFC(1),M)
      IF(ID.EQ.1) GO TO 430
      DO 380 I=1,K0
```

```
380 IFR(I)=0
    N=0
    NCS=(-1)*IS
395 N=N+1
    IF(IFR(N).NE.0) GO TO 395
    NCS=NCS+IS
    IFR(N)=NCS+1
    DO 420 I=2,IS
420 IFR(MOD(N*IFC(I-1),M))=NCS+I
    IF(NCS+IS.LT.K0) GO TO 395
430 IND=7*IDD+IC
    GO TO(445,455,465,445,445,455,445,310,310,310,455,465,310,455,310,
   1310,310,310,310,465,465,310,310,310,310,310,310,310),IND
445 IDD=1
    GO TO 470
455 IDD=2
    GO TO 470
465 IDD=3
470 CALL SECOND(TI)
    IF(IDD.EQ.1) GO TO 495
    DO 490 I=1,K0
    IF(IDD.EQ.2) IR(I)=IFR(I)
490 IF(IDD.EQ.3) IR(IFR(I))=IA*(I+IB-1)
495 M3=NCP*(4-NCP/35)+3
    K1=NCP
    I=0
510 I=I+1
    IF(I.GT.NPW) GO TO 715
    K3=I*NCP
    K2=K3-NCP+1
530 J=M2=0
535 J=J+1
    IF(J.GT.NPL) GO TO 695
    M2=J*NRP
    M1=M2-NRP+1
555 IF(IDD.EQ.1) WRITE(2,170) M,IS
    IF(IDD.EQ.2) WRITE(2,180) M,IS
    IF(IDD.EQ.3) WRITE(2,190) M,IS
    IF(NCP.EQ.29) WRITE(2,200) F,(G,K,K=K2,K3)
    IF(NCP.EQ.39) WRITE(2,205) F,(G,K,K=K2,K3)
    WRITE(2,210) (H,K=1,M3)
    DO 685 L=M1,M2
    IF(IDD-2) 595,625,650
595 IR(K2)=MOD(K2*IFC(L),M)
    IF(K2.EQ.K3) GO TO 680
    K21=K2+1
    DO 615 K=K21,K3
615 IR(K)=MOD(IR(K-1)+IFC(L),M)
    GO TO 680
625 IF(L.EQ.1) GO TO 680
    DO 640 K=K2,K3
    IR(K)=IR(K)+1
640 IF(MOD(IR(K)-1,IS).EQ.0) IR(K)=IR(K)-IS
    GO TO 680
650 IF(L.EQ.1) GO TO 680
    DO 675 KK=IS,K0,IS
    IRR=IR(KK)
    DO 670 K=2,IS
670 IR(KK+2-K)=IR(KK+1-K)
```

```
675 IR(KK-IS+1)=IRR
680 IF(NCP.EQ.29) WRITE(2,215) L,F,(G,IR(K),K=K2,K3)
685 IF(NCP.EQ.39) WRITE(2,220) L,F,(G,IR(K),K=K2,K3)
    GO TO 535
695 IF(M2.EQ.IS) GO TO 510
    M2=IS
    M1=IS-NRR+1
    GO TO 555
715 IF(K3.EQ.K0) GO TO 745
    K1=NRC
    K2=NPW*NCP+1
    K3=K0
    M3=NRC*(4-NCP/35)+3
    GO TO 530
745 IF(IDD-2) 775,770,750
750 IF(IA.EQ.-1) WRITE(2,225) E
    IF(IA.EQ.1) WRITE(2,225) H
    WRITE(2,260) IB
    GO TO 775
770 WRITE(2,230)
775 CALL SECOND(TJ)
    TK=TJ-TI
    WRITE(2,290) TK
    K3=0
    GO TO 430
170 FORMAT(1H1,17X,'I N I T I A L   A R R A Y   O F   M O D U L O   M
   1=',I4,'   A N D   O R D E R   S =',I4/1H ,17X,85('-')/)
180 FORMAT(1H1,16X,'C O D I N G   A R R A Y   O F   M O D U L O   M ='
   1,I4,'   A N D   O R D E R   S =',I4/1H ,18X,83('-')/)
190 FORMAT(1H1,16X,'S Y N D R O M E   A R R A Y   O F   M O D U L O
   1M =',I4,'   A N D   O R D E R   S =',I4/1H ,16X,87('-')/)
200 FORMAT(1H0,3X, A1/1H+,3X,29(A1,I3))
205 FORMAT(1H0,3X, A1/1H+,3X,39(A1,I2))
210 FORMAT(1H ,120A1)
215 FORMAT(1H ,I3, A1/1H+,3X,29(A1,I3))
220 FORMAT(1H ,I3, A1/1H+,3X,39(A1,I2))
225 FORMAT(/1H0,1X,116('-')/1H ,69X,A1)
230 FORMAT(/1H0,16X, 88('-')/1H ,16X,'FOR AN (N*(K0+IS),N*K0) ''CYCLIC
   1'' C.S.O.C., THE ITH PARITY-CHECK DIGIT OF BLOCK C, IS GI-'/1H ,16
   2X,'VEN BY THE MODULO-2 SUM OF MESSAGE DIGITS M(C-J+1,JAR(I,J))/J=1
   3,2,...,N, FOR ALL I=1,2,.'/1H ,16X,'.,IS, WHERE M(L,I) DENOTES THE
   4 ITH MESSAGE DIGIT OF BLOCK L AND JAR IS THE CODING ARRAY.'/1H ,16
   5X, 88('-'))
260 FORMAT(1H+,' FOR AN (N*(K0+IS),N*K0) ''CYCLIC'' C.S.O.C., SYNDROME
   1 DIGITS SYN(I,C-B KAR(I,J))/I=1,2,...,IS ARE ORTHOGONAL ON THE JTH'
   2/1H ,' DIGIT OF BLOCK C, WHERE: SYN(I,K) DENOTES THE ITH SYNDROME
   3DIGIT OF BLOCK K, KAR IS THE SYNDROME ARRAY, AND B=',I5,'.'/1H ,1X
   4,116('-'))
290 FORMAT(/1H0,20X,'PROCESSING TIME =',1PE9.2/1H0,120('='))
    END
```

Figure A8.2.8: FORTRAN programme for *subroutine CODAR1.*

Subroutine CODAR3 returns enough information about the EA & SA to enable the decoder to operate, without requiring the use of large storage-arrays. It is used for the simulation

```
      SUBROUTINE COTAF3(KO,IS,JR,JS,LUC,IND,IFB,MAA,MAB,MAC,KR)
      DIMENSION JR(KO),JS(KO),LUC(KO),IND(KO),IFB(KO),MAA(KO),MAB(KO),
     1 MAC(KO),KR(KO)
      COMMON/COF2/JEXP/MAIN/LE
      XLE=LE
      M=KO+1
      LTR=KR(1)
      KR(1)=JEXP=IEXP1(K,IS)
      KR(1)=JEXP=4
      IF(JEXP.EQ.0) RETURN
      DO 100 I=1,KO
  100 JR(I)=0
      NCS=1
      NCC=IS+2
      N=0
  110 N=N+1
      IF(JR(N).NE.0) GO TO 110
      JR(N)=LLC=AMOD(NCS-0.5,XLE)+0.5
      JS(N)=(NCS-0.5)/LE+1
      LUC(N)=LU=AMOD(NCC-2.5,XLE)+0.5
      IND(N)=IC=ISIGN(1,LLC-LU)+2
      FCM=IC+2.5-1/IC*2
      KR(NCS)=N-K
      KR=1
      DO 120 I=2,IS
      KR=MOD(KK+JEXP,KR)
      K=MOD(N+KR,M)
      JR(K)=JJ=AMOD(NCS+I-1.5,XLE)+0.5
      JS(K)=(NCS+I-1.5)/LE+1
      LUC(K)=LU
      IND(K)=FCM+SIGN(1.5,LU-JJ+0.5)-1/JJ*2
  120 KR(NCC-1)=N-K
      NCS=NCS+IS
      NCC=NCC+IS
      IF(NCS.LT.M) GO TO 110
      MA1=COMPL(MASK(LE-IS))
      DO 130 I=1,KO
      IFB(I)=MAB(I)=MAC(I)=0
      MAA(I)=MA1
      GO TO(130,140,150,160,170,180),IND(I)
  140 IFB(I)=JR(I)-LR-1
      MA=MASK(LE+LUC(I)+1-JR(I)-IS)
      MAA(I)=AND(MA,MA1)
      MAB(I)=COMPL(MA)
      GO TO 130
  150 IFB(I)=LUC(I)-LE
      MA=MASK(LUC(I))
      MAA(I)=AND(MA,MA1)
      MAB(I)=COMPL(MA)
      GO TO 130
  160 MA=MASK(IS-LUC(I))
      MAA(I)=AND(MA,MA1)
      MAB(I)=COMPL(MA)
      GO TO 130
  170 IFB(I)=LU=JR(I)+IS-2*LE-1
      MC=MASK(LUC(I)-LU)
      MB=COMPL(MASK(LE-LUC(I)))
```

```
      MF(I)=SHIFT(ME,LF+LU-LUC(I))
      MA(I)=AND(MC,MA1,COMPL(MAF(I)))
      MC(I)=COMPL(MC)
      GO TO 13.
120 IFE(I)=JP(I)-1
      C=MASK(LE-IFE(I))
      ME=COMPL(MASK(LE+LUC(I)-IS))
      MAB(I)=SHIFT(ME,IFE(I))
      MA(I)=AND(MA1,MC,COMPL(MAB(I)))
      MAC(I)=COMPL(MC)
13    CONTINUE
      IF(LI-.EQ.0) RETURN
      WRITE(2,500) N,IS
      WRITE(2,510)
      WRITE(2,520)
      WRITE(2,530)
      WRITE(2,540)
      WRITE(2,550)
      WRITE(2,560)
      WRITE(2,570)
      WRITE(2,580)
      WRITE(2,590)
      WRITE(2,600)
      WRITE(2,610)
      WRITE(2,620)
      WRITE(2,630)
      JCF=IS-LE-1
      DO 200 I=1,NO
      IFA=JP(I)+JCF
      IFC=(JP(I)-1-IF)*(IND(I)/5)
      CALL BITOUR(MAA(I),L-,I1,I2,I3)
      CALL BITOUR(MAB(I),LE,J1,J2,J3)
      CALL BITOUR(MAC(I),L-,K1,K2,K3)
200 WRITE(2,640) JP(I),JS(I),LUC(I),IND(I),IFA,IFB(I),IFC,I1,I2,I3,J1,
     1J2,J3,K1,K2,K3,KA(I)
      RETURN
C     FORMAT(1H1,4X,'C O D I N G   &   S Y N D R O M E   A R R A Y S   O
     1 F   M O D U L O   N =',I5,'   A N D   O R D E R   I S =',I3/1H ,
     24X,111('-')/1H0,13X,'MOD-2 SUM OF [BIT JP(I) IN WORD IRA(JS(I),I)]
     3/I=1,2,....,K  =  PARITY BIT 1, (MSB=1,LSB=LB)'/1H ,  5X,'IN EACH
     4 SUB-BLOCK BITS THAT CONTRIBUTE TO CURRENT PARITY APPEAR IN A GROUP
     5 OF (IS) CONSECUTIVE BIT POSITIONS'/1H0,'EACH GROUP IS DIVIDED IN
     6 AT MOST 3 PARTS:PART A FROM BIT POSN JP(I) OF IRA(JS(I),I) TO RIGH
     7TMOST BIT POSN OF GROUP, IN'/1H ,'THE SAME WORD - PART B FROM 1ST
     8 BIT POSN OF GROUP IN NEXT TO THE RIGHT OR NEXT TO THE LEFT WORD...
     9.ETC'/1H ,'FOR EACH SUB-BLOCK THERE ARE 6 STATES: STATE 1 HAS 1 PA
     ART - STATES 2,3 & 4 HAVE 2 PARTS, IN 2 BOTH IN THE SAME WORD AND'/
     B1H ,'IN 3 & 4 IN DIFFERENT WORDS (IN 3 PART A IS LEFT PART OF GROU
     CP IN 4 THE OPPOSITE) - STATES 5 & 6 HAVE 3 PARTS, A & C'/1H ,'BEIN
     DG IN THE SAME WORD (IN 5 PART C IS LEFT PART OF GROUP, IN 6 PART B
     E IS LEFT).-'/1H ,'TO DECODE BIT (N-1)*IS+1/N=1,2,....,KC/IS ,  COLL
     FECT SYNDROME BITS IN POSN (LB+J-IS) OF SYN(KP((N-1)*IS+J))/J=1,2,'
     G/1H ,'....,IS ,  WHERE SYN(KP) IS THE CURRENT SYNDROME,  MSB=1,LSB=
     HLB,  AND THE REST BITS ARE DECODED BY APPROPRIATE SHIFTS.-'//)
510 FORMAT(1H0,'ARRAY JP = [BIT POSITION OF 1ST OF GROUP]')
520 FORMAT(1H ,'ARRAY JS = [WORD OF 1ST OF GROUP]')
530 FORMAT(1H ,'ARRAY LUC = [POSITION OF RIGHTMOST BIT IN GROUP]')
```

```
54   FORMAT(1H ,'ARRAY IND = [STATE OF GROUP]')
55   FORMAT(1H ,'ARRAY IFA = [SHIFT OF PART A]')
560  FORMAT(1H ,'ARRAY IFB = [SHIFT OF PART B]')
570  FORMAT(1H ,'ARRAY IFC = [SHIFT OF PART C]')
580  FORMAT(1H ,'ARRAY MAA = [MASK FOR PART A]')
590  FORMAT(1H ,'ARRAY MAB = [MASK FOR PART B]')
6    FORMAT(1H ,'ARRAY MAC = [MASK FOR PART C]')
61   FORMAT(1H ,'ARRAY KR = [SYNDROME POSITIONS]')
620  FORMAT(1H0,'MASK ARRAYS ARE GIVEN AS 3-TUPLES: (I1,I2,I3)/ WHERE:
    1I1(I3) = NO OF LEFT(RIGHT)MOST 0''S & I2 = NO OF MIDDLE 1''S')
63   FORMAT(///1H ,'   JP    JS    LOC    IND    IFA    IFB    IFC',7X,'MAA',
    110X,'MAB',10X,'MAC',6X,'KR')
640  FORMAT(1H ,2I5,I6,I5,I7,2I6,3(3X,'(',2(I2,','),I2,')'),I5)
     END
```

<u>Figure A8.2.9</u>: FORTRAN programme for *subroutine CODAR3.*

of very long codes (with a k of the order of 1,000). This subroutine (see Fig. A8.2.9) is used by the simulation programme *IKOSI5* (see Fig. A8.4.2), as well as by other main programmes.

All three routines were tested for moduli up to 1500.

<u>Example A8.2.1</u>: Let the (18,6) type-C5 code of Example A7.12.4 (p. 493). From the IA, the k/J = 3 coset leaders are C_r = 1,2 & 4. Then, from (A8.1.3):

$Jarr(i,C_r) = C_n + (i-1)18/6 \quad /i=1,2,\ldots,6 \quad \longrightarrow$

$Jarr(i,1) = 1 + 3(i-1) = 1,4,7,10,13,16$
$Jarr(i,2) = 2 + 3(i-1) = 2,5,8,11,14,17$
$Jarr(i,4) = 3 + 3(i-1) = 3,6,9,12,15,18$

From (A8.1.4), the columns corresponding to coset leader C_r are given by [$Karr(i,1) = a_{i,1}$ = 1st column of the IA]:

$Cl_n(i) = C_r \times a_{i,1} \bmod k+1 \quad /i=1,2,\ldots,J \quad \longrightarrow$

$Cl_1(i) = 1 \times (8,7,18,11,12,1) \bmod 19 = 8,7,18,11,12,1$

$Cl_2(i) = 2 \times (8,7,18,11,12,1) \bmod 19 = 16,14,17,3,5,2$

$Cl_3(i) = 4 \times (8,7,18,11,12,1) \bmod 19 = 13,9,15,6,10,4$

The last row of Jarr is given by (A8.1.5):

$Jarr(6,Cl_n(i)) = C_n + (i-1)18/6 \quad \longrightarrow$

$Jarr(6,Cl_1(i)) = 1+3(i-1) \longrightarrow$

$$Jarr(6,\{8,7,18,11,12,1\}) = 1,4,7,10,13,16$$

$Jarr(6,Cl_2(i)) = 2+3(i-1) \longrightarrow$

$$Jarr(6,\{16,14,17,3,5,2\}) = 2,5,8,11,14,17$$

$Jarr(6,Cl_3(i)) = 3+3(i-1) \longrightarrow$

$$Jarr(6,\{13,9,15,6,10,4\}) = 3,6,9,12,15,18$$

The three expressions above give the last row of Jarr (the EA). The element of the first row will be the element of the last plus k/J = 3 (minus k=18, if it exceeds 18):

1 2 14 3 17 15 7 4 9 18 13 16 6 8 12 5 11 10

and then the EA (Jarr) will be:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	2	14	3	17	15	7	4	9	18	13	16	6	8	12	5	11	10
4	5	17	6	2	18	10	7	12	3	16	1	9	11	15	8	14	13
7	8	2	9	5	3	13	10	15	6	1	4	12	14	18	11	17	16
10	11	5	12	8	6	16	13	18	9	4	7	15	17	3	14	2	1
13	14	8	15	11	9	1	16	3	12	7	10	18	2	6	17	5	4
16	17	11	18	14	12	4	1	6	15	10	13	3	5	9	2	8	7
1	2	2	3	2	3	1	1	3	3	1	1	3	2	3	2	2	1

Finally, from (A8.1.8), $Karr(j,Jarr(j,i)) = 19-i$, for $i=1,2,3,\ldots,18$ & $1 \le j \le 6$, and the SA (Karr) is:

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
18	17	15	11	3	6	12	5	10	1	2	4	8	16	13	7	14	9
7	14	9	18	17	15	11	3	6	12	5	10	1	2	4	8	16	13
8	16	13	7	14	9	18	17	15	11	3	6	12	5	10	1	2	4
1	2	4	8	16	13	7	14	9	18	17	15	11	3	6	12	5	10
12	5	10	1	2	4	8	16	13	7	14	9	18	17	15	11	3	6
11	3	6	12	5	10	1	2	4	8	16	13	7	14	9	18	17	15

So to decode, say, $e_h^{(5)}$ syndromes $ISR(Karr(j,5),j)$ $/j=1,2,\ldots,6$ are needed, i.e. $ISR(\{3,17,14,16,2,5\},j)$ $/j=1,2,\ldots,6$. Using (A8.1.7), $ISR(z,j) = s^{(j)}_{h+18-z}$ $/j=1,2,\ldots,6$ & $z=1,2,\ldots,18$, syndrome bits $s^{(1)}_{h+15}$, $s^{(2)}_{h+1}$, $s^{(3)}_{h+4}$, $s^{(4)}_{h+2}$, $s^{(5)}_{h+16}$ & $s^{(6)}_{h+13}$ check on $e_h^{(5)}$. To verify this, from Theorem 7.1, syndrome bits $s^{(x)}_{h+w-1}$ check on $e_h^{(5)}$, for all $b_{x,w} = 5$. From the EA above, $b_{1,16} = b_{2,2} = b_{3,5} = b_{4,3} = b_{5,17} = b_{6,14} = 5$, hence $s^{(1)}_{h+15}$, $s^{(2)}_{h+1}$, $s^{(3)}_{h+4}$, $s^{(4)}_{h+2}$, $s^{(5)}_{h+16}$ & $s^{(6)}_{h+13}$, should check on $e_h^{(5)}$.

∎

A8.2.9. Effective Constraint-Length

```
      FUNCTION NEFEL1(M,IS)
      LOGICAL ISU(999)
      COMMON/COD2/JEXP
      K0=M-1
      IA=JEXP
      NEFEL1=0
      IF(IA.EQ.0) RETURN
      IF(MOD(IS,2).EQ.1.AND.IS.GT.4) GO TO 230
      NEFEL1=(IS+1)/2*M+1
      RETURN
  230 DO 240 I=1,K0
  240 ISU(I)=.FALSE.
      DO 340 N=1,K0
      IF(ISU(N)) GO TO 340
      JA=N
      JSU=1
      DO 320 I=1,IS
      JA=MOD(JA*IA,M)
      ISU(JA)=.TRUE.
  320 JSU=JSU+JA
      NEFEL1=MAX0(NEFEL1,JSU)
  340 CONTINUE
      RETURN
      END
```

Figure A8.2.10: FORTRAN programme for *function NEFEL1*.

APPENDIX 8.3: CHANNEL AND DECODER SIMULATION

The majority-logic decoder of Fig. 5.1 is used as a model. The decoder has to store one constraint-length of received message bits.* This is done in k x k array IRA, with the currently received block stored in the first row:

$$IRA(z,i) = r^{(i)}_{h+1-z} \quad /i=1,2,\ldots,k \ \ \& \ \ z=1,2,\ldots,k \qquad (A8.3.1)$$

It is also necessary to store the currently received block of parity-checks. This is done in 1 x J array JRA:

$$JRA(j) = r^{(k+j)}_{h} \quad /j=1,2,\ldots,J \qquad (A8.3.2)$$

The other array needed is the syndrome register. This was defined earlier by (A8.1.7), assuming that the currently decoded block is the hth. Since, now, the currently received block is the hth,

$$ISR(z,j) = s^{(j)}_{h+1-z} \quad /j=1,2,\ldots,J \ \& \ z=1,2,\ldots,k \qquad (A8.3.3)$$

* See Appendix 8.4 (§ A8.4.1., p. 541), for the corresponding FORTRAN programme.

The decoder processes one block at a time. The first operation required, is the shifting of arrays IRA & ISR downward by one row, to make space for the current blocks, $r_h^{(m)}$ & s_h. Subsequently, $r_h^{(m)}$ & $r_h^{(p)}$ are stored in IRA(1,i) & JRA. The next operation is the collection of statistical results about the channel (number of channel errors). Following that, the current syndrome block must calculated. From eqns (7.5), (A8.3.1), (A8.3.2) & (A8.3.3):

$$ISR(1,j) = \sum_{z=1}^{k} IRA(z,Jarr(j,z)) + JRA(j) \quad /j=1,2,..,J \qquad (A8.3.4)$$

Normally, z ranges from 1 to MIN{k,h}. This may be simplified to [1,k] if IRA is initialized, prior to the reception of the 1st block.

The next step is the addition* of the syndrome bits checking on each of $e^{(i)}_{h-(k-1)}$ /i=1,2,...,k. ISR(Karr(j,i),j) /j=1,2,...,J are the syndrome bits checking on $e^{(i)}_{h-(k-1)}$ /i=1,2,...,k [see (A8.1.8)]. Hence, their sum*, Ipcs(i), is

$$Ipcs(i) = \sum_{j=1}^{J} ISR(Karr(j,i),j) \quad /1 \le i \le k \qquad (A8.3.5)$$

If Ipcs(i) > T, then $\tilde{e}^{(i)}_{h-(k-1)} = 1$ (see Theorem 5.3). After the decoding of the k bits, the number of decoding errors in that block is obtained. Finally, for the case of feedback decoding, the syndrome register ISR is reset. If the estimated error bit is 1, then the syndrome bits that were used for its estimation are inverted. From (A8.3.5), these bits are ISR(Karr(j,i),j) /j=1,2,...,J.

Fig A8.3.1 shows the flow-chart of the channel simulator and the decoder. Nce is the number of channel errors, Σ1 will be used to estimate $E[n_c]$ (expected to be 0) and Σ2 will be used to estimate $E[n_c^2]$ (expected to be σ^2).

The above-described technique is not memory-efficient with very long codes. The reason is that one bit is stored in one word, which can store, say, b bits (b=60, for the mainframe computer used). The total memory-requirement for arrays IRA, ISR & JRA, is $k^2+(k+1)J$. If bit-manipulation routines are used, then the total memory requirement may be

* Arithmetic.

```
┌─> For i=1,2,...,k
│   n_c = G05DDF(0,σ)   &   If (n_c > 0.5), then:  e = 1
│                           If (n_c ≤ 0.5), then:  e = 0
└─  IRA(1,i) = e  &  Nce = Nce+e  &  Σ1 = Σ1+n_c  &  Σ2 = Σ2+n_c²
┌─> For i=1,2,...,J
│   n_c = G05DDF(0,σ)   &   If (n_c > 0.5), then:  e = 1
│                           If (n_c ≤ 0.5), then:  e = 0
└─  JRA(i) = e    &  Nce = Nce+e  &  Σ1 = Σ1+n_c  &  Σ2 = Σ2+n_c²

┌─> Step:  h = h+1*
│ ┌─> For i=2,3,...,k
│ │   IRA(k+2-i,j) = IRA(k+1-i,j)   /j=1,2,...,k
│ └─  ISR(k+2-i,j) = ISR(k+1-i,j)   /j=1,2,...,J
│    Calculate the current syndrome block, from (A8.3.4),
│    and store it in ISR(1,i)
├─── If (h < k), then: Repeat step
│    If else, then:
│ ┌───────────────────> For i=1,2,...,k
│ │                     Calculate IPCS(i), from (A8.3.5)
│ │                     If [Ipcs(i) > T], then:   Invert IRA(k,i)
│ │                     Update decoding-errors register
│ ├──────────────────── If [no FD, or Ipcs(i) ≤ T], then:  next i
│ │                     If else, then:
│ └──────────────────── Invert syndromes ISR(Karr(j,i),j)  /j=1,2,...,J
└── Repeat step
```

Figure A8.3.1: Flow-chart for the decoding of type-C5 codes.

reduced by a factor of, about, b permitting thus $\sqrt{b}$ longer codes to be tested. Since b was adequately large, it was decided to restrict $J \le b$. Then array JRA becomes a variable, while array ISR a $1 \times k$ array. The current J received parity-checks are stored in the last J least-significant bit (LSB) positions, while the current syndrome block is stored in the J LSB positions of ISR(1). IRA is a $k_b \times k$ array organized differently: The current block is stored in the first column, which has $b \times k_b$ bit positions, where $b \times k_b \ge k$, or $k_b \ge k/b$, or $k_b = \lfloor (k-0.5)/b \rfloor + 1$. So if $k = q \times b$, then $k_b = q$, while if $k = q \times b + 1$, $k_b = q+1$. The first bit of the current block is stored in the most-significant bit (MSB) position.

The shift of IRA & ISR is simpler than before. The next operation is the formation of the current received message block (to be stored in the 1st column of IRA). This is done via the bit-manipulation functions MASK, SHIFT & OR. The basic problem is the formation of a (b-bit) word, say, W which contains the first b bits of the received (k-bit) message block. MAS1 = MASK(1) is a word with 1 in the 1st position and 0s in the rest. Assuming that W has been initial-

* h is the currently received block.

ized to W=0, W = OR(W,MAS1) will store a 1 in the first position of W. Hence, if the 1st bit is 1, W = OR(W,MAS1). MAS1 = SHIFT(MAS1,1) is a word with a 1 in its 2nd position and 0s in the rest of its positions. Then, if the 2nd bit is 1, W = OR(W,MAS1), etc. After b iterations, W contains the first b bits and IRA(1,1) = W. This is repeated for the subsequent group of b bits, until all the k received message bits have been stored in IRA(i,1). The same technique is used for the received parity-check bits.

The next operation is the calculation of the current syndrome block. This requires the EA which is returned by subroutine CODAR2, for the decoder implementation of Fig. A8.3.1. While CODAR2 requires a total of 2 × J × k words of memory, subroutine CODAR3 (see § A8.2.8., p. 529) is used for the 'long' codes, both because it returns all the information necessary to implement the various bit-manipulation operations and also because it requires a total of 9×k words of memory, for the various arrays. The following example will explain the technique used for syndrome calculation.

<u>Example A8.3.1</u>: Let the (12,4) type-C5 code, and its encoding array (EA)*:

	1	5	8	9	2	12	10	4	11	6	7	3
	2	6	5	10	3	9	11	1	12	7	8	4
	3	7	6	11	4	10	12	2	9	8	5	1
	4	8	7	12	1	11	9	3	10	5	6	2
Column No:	1	2	3	4	5	6	7	8	9	10	11	12
Coset No:	1	2	2	3	1	3	3	1	3	2	2	1

Array IRA has dimensions $k_b \times k$. Assume that b=7, in this case. Then, $k_b = \lfloor(12-0.5)/7\rfloor+1 = 2$, so IRA is 2 × 12. Its 'bit-structure' is shown in Figure A8.3.2; symbols 'x','o','+' & '#' denote the received bits participating in the formation of the jth current syndrome bit /j=1,2,3,4, respectively. The mod-2 sum of all the xs will give the 1st syndrome bit (minus the 1st current received parity-check). So, what is required is the generation of k (=12) words, each corresponding to a different column of IRA, with the J (=4) received bits in the last J (=4) least significant bit positions, in order x o + #. Then the XOR sum of these k (=12) words will contain the J (=4) current syndromes (minus

* The codes were simulated using McQuilton's mapping - see discussion following Definition 7.7, p. 220.

the parity-checks), in its last J (=4) least significant bit positions. The XOR sum of this word with JRA equals the current syndrome block ISR(1).

Note, from Fig. A8.3.2, that the bits that are used for the calculation of the current syndrome block appear in groups of J (=4), which are, also, cyclic shifts of each other. What is required, for each of the k (=12) groups, is to shift the J bits so that they occup the last J LSB positions of a word, W: W = [????xo+#].

1	2	3	4	5	6	7	8	9	10	11	12	<— Word	
x				#			o				+	1	<— Bit position
o				x			+				#	2	First Word
+				o			#				x	3	
#				+			x				o	4	
	x	o							#	+		5	
	o	+							x	#		6	
	+	#							o	x		7	
	#	x							+	o		1	Second Word
			x		o	#		+				2	
			o		+	x		#				3	
			+		#	o		x				4	
			#		x	+		o				5	
												6	
												7	

1	5	1	2	2	5	3	4	4	6	7	3	JR (bit posit. of x)
1	1	2	2	1	2	2	1	2	1	1	1	JS (word of x)
4	1	1	5	4	5	5	4	5	1	1	4	LUC (posit. of last bit)
1	3	4	1	2	2	2	2	2	5	5	2	IND (state of group)
-3	1	-3	-2	-2	1	-1	0	0	2	3	-1	IFA (shift of part A)
	-6	0		-6	-3	-5	-4	-4	-5	-4	-5	IFB (shift of part B)
									-2	-1		IFC (shift of part C)
4	3	1	4	3	1	3	1	2	2	1	2	NBA (No of bits in part A)
	1	3		1	3	1	3	2	1	1	2	NBB (No of bits in part B)

Figure A8.3.2: Organization of IRA, for the (12,4) code.

Note that each group is partitioned into two parts because the first bit of the group ('x') is not always leading (as in columns 1,2 & 4). Furthermore, there are cases where a group extends over two different words (as in columns (2,3,10 & 11). Because the length of a group is J and the length of a word has been taken to be at least J, then one may partition each group into up to 3 parts. Each part is located in a single word and contains as many of the J bits in the right sequence (x o + #); part A starts with 'x'. If $G_X(i)$ /X=A,B,C & i=1,2,...,k denotes part X of the ith group (column i of IRA), then: $G_A(1)$ = [x o + #], $G_A(2)$ = [x o +], $G_A(6)$ = [x], $G_A(10)$ = [x o], etc. $G_B(1)$ = [ ], $G_B(2)$ = [#], $G_B(3)$ = [o + #], $G_B(10)$ = [+], $G_B(11)$ = [o], etc. Finally all but groups 10 & 11 do not have part C; $G_C(10)$ = [#] & $G_C(11)$ = [+ #]. Subroutine CODAR3 returns, in arrays JR, JS, LUC & IND, information about each group. For the ith group, JR(i) is the bit position of 'x', JS(i) is the word where 'x' belongs, LUC(i) is the bit position of the last bit of the group and IND(i) is the state of the group (see Fig. A8.3.2). A group is, in state 1 if it is made of one part, in state 2 if it is made of two parts both in the same word, in state 3 if it is made of two parts of which B is in the 2nd word, in state 4 if it is made of two parts of which A is in the 2nd word, in state 5 if it is made of 3 parts of which B is in the 2nd word and in state 6 if it is made of 3 parts of which C & A are in the 2nd word (see Fig. A8.3.2).

Arrays IFA, IFB & IFC contain information about the shift required for each part of the group, so that they are shifted in the right bit position within a word (for 'x' it is bit position 4, for 'o' 5, for '+' 6 & for '#' 7). Nevertheless, CODAR3 returns only IFB; IFA is readily obtained from IFA(i) = JR(i)+J-b-1 (b=7, here), and IFC from IFC(i) = JR(i)-b-1 [for IND(i) = 5, or 6].

Finally, a mask is required for each part of each group, such that it contains 1s only at the bit positions of the shifted part and 0s elsewhere. The information provided by arrays NBA & NBB helps build this mask. For example, NBA(2) = 3 (part A of group 2 is made of 3 elements - x o & +). Once shifted in its appropriate bit position (4,5,6), part A requires mask (0001110), while part B requires mask (0000001). In general, if part A is made of NBA(i) bits then MAA(i), the mask for part A, contains NBA(i) consecutive 1s. The first 1 should be in the position where 'x' will reside, which is b-J+1. So, MAA(i) = [b-J,NBA(i),J-NBA(i)], i.e. MAA(i) is made of b-J 0s (starting from the MSB), followed by NBA(i) 1s, followed by J-NBA(i) 0s. The mask for part B, MAB(i), contains NBB(i) 1s. The first element of part B is the [NBA(i)+1]th of the group, which will be shifted to position b-J+NBA(i)+1, hence MAB(i) is made of b-J+NBA(i) 0s, followed by NBB(i) 1s, followed by 0s: MAB(i) = [b-J+NBA(i),NBB(i),J-NBA(i)-NBB(i)]. The mask for part C contains J-NBA(i)-NBB(i) 1s. The first element of part C is the [NBA(i)+NBB(i)+1]th of the group, which will be shifted to position b-J+NBA(i)+NBB(i)+1, hence: MAC(i) = [b-J+NBA(i)+NBB(i),J-NBA(i)-NBB(i),0].

Consider now group 10. From Fig. A8.3.2, IND(10)=5, hence this is made of 3 parts, of which B is in the 2nd word, hence the group format is C | A ‖ B (had it been A | C ‖ B, then it would have been made of two parts, A ‖ B). The first operation is the shifting of its 3 parts. Parts A & C are in IRA(JS(10),10), i.e. in the 1st word of the 10th column of IRA, while part B in IRA(JS(10)+1,10). According to the above, IFA(10) is calculated by IFA = JR(i)+J-b-1 = 6+4-7-1 = 2 and IFC by IFC(10) = JR(i)-b-1 = 6-7-1 = -2. Hence:

KRA = SHIFT(IRA(JS(i),i),IFA) = SHIFT(IRA(JS(10),10),2) =
    = SHIFT(IRA(1,10),2) = [??#xo??]
KRB = SHIFT(IRA(JS(i)+1,i),IFB(i)) =
    = SHIFT(IRA(JS(10)+1,10),IFB(10)) = SHIFT(IRA(2,10),-5)
    = [?????+?]
KRC = SHIFT(IRA(JS(i),i),IFC) = SHIFT(IRA(JS(10),10),-2) =
    = SHIFT(IRA(1,10),-2) = [??????#]

The masks for the three parts are:

MAA(i) = [b-J,NBA(i),J-NBA(i)] = [3,2,2] = [0001100]
MAB(i) = [b-J+NBA(i),NBB(i),J-NBA(i)-NBB(i)] = [5,1,1] =
       = [0000010]
MAC(i) = [b-J+NBA(i)+NBB(i),J-NBA(i)-NBB(i),0] = [6,1,0] =
       = [0000001]

Using the masks:

KRA = AND(KRA,MAA(10)) = AND([??#xo??],[0001100]) =
    = [000xo00]
KRB = AND(KRB,MAB(10)) = AND([?????+?],[0000010]) =
    = [00000+0]
KRC = AND(KRC,MAC(10)) = AND([??????#],[0000001]) =

= [000000#]

Finally:

W = XOR(KRA,KRB,KRC) = XOR([000xo00],[00000+0],[000000#]) =
= [000xo+#]

The above is repeated for all k groups. The Ws are added mod-2 to JRA. The final result is ISR(1), the current syndrome block.

The next step, in the decoding of 'long' codes, is the estimation of the k error bits. The SA provides the information about the syndromes checking on each error bit. To economize on storage space, only the 'equivalent' of one row of the SA is returned by CODAR3, the rest of the rows being generated during decoding. Consider the example below:

<u>Example A8.3.2</u>: Let the (12,4) code discussed in Example A8.3.1. From its EA, and eqn (A8.1.8), Karr(j,Jarr(j,i)) = k+1-i /i=1,2,...,k & 1≤j≤J:

12	8	1	5	11	3	2	10	9	6	4	7
5	12	8	1	10	11	3	2	7	9	6	4
1	5	12	8	2	10	11	3	4	7	9	6
8	1	5	12	3	2	10	11	6	4	7	9
1	2	3	4	5	6	7	8	9	10	11	12

Note that the SA is partitioned in 3 groups (one for each coset). Within a coset, column i is a downward cyclic shift by one, of column i-1. Hence, if CODAR3 returns the 1st column of each coset, the rest are easily generated.

Let 1 x k array KR be:

KR(i) = 12 5 1 8 11 10 2 3 9 7 4 6

for i=1,2,...,12, respectively. To decode the ith bit, one would determine the coset number, C_n, first, by letting $C_n = \lfloor(i-0.5)/J\rfloor+1$. If i=7, $C_n = \lfloor 6.5/4\rfloor+1 = 2$. This means that the appropriate KR elements are 11 10 2 3. Thereafter, the relative shift within this group is determined by i mod J = 7 mod 4 = 3. Hence, the SA column for i=7 is 2 3 11 10. ISR contains the J (=4) syndrome bits, checking on the ith (7th) error bit, in rows 2, 3, 11 & 10, and in bit positions

4, 3, 2 & 1, respectively, counting from the LSB. Then, the sum of the J (=4) syndromes is obtained as following:

```
IPCS = 0
 JSR = SHIFT(ISR(2),3)
 JSR = AND(JSR,1)
IPCS = IPCS+JSR
 JSR = SHIFT(ISR(3),2)
 JSR = AND(JSR,1)
IPCS = IPCS+JSR
    etc
```

IPCS is compared with T for the final decision.

The syndrome-register feedback is done in the same way. This time, though, some masks must be used in order to invert only the appropriate bit from each of the J rows of the ISR that contain syndromes checking on the particular error bit.*

APPENDIX 8.4: SIMULATION PROGRAMMES

A8.4.1. Software Implementation of the Decoder

The FORTRAN software in Fig. A8.4.1, is the part of the computer simulation-programme that processes one block of

```
      [illegible]
C  INITIALIZATIONS.-
      NP=NM=NW(1)=NW(2)=JCDS=NOB=0
      PCS1=PCS2=AF=0.0
      CALL G05CBF(KG)
      DO 120 I=1,K0
      DO 130 J=1,K0
  130 IRA(I,J)=0
      DO 120 J=1,IS
  120 ISR(I,J)=0
      DO 125 I=1,IS
  125 JRA(I)=0
      DO 150 I=1,NAP
  150 AFD(I)=0
C  MAIN LOOP.-
      IJ=0
  145 IJ=IJ+1
```

* See Appendix 8.4 (§ A8.4.2., p. 543), for the corresponding FORTRAN programme.

```
C  SHIFT.-
      DO 170 I=2,K0
      K=K0+2-I
      DO 160 J=1,K
  160 IRA(K,J)=IRA(K-1,J)
      DO 170 J=1,IS
  170 ISR(K,J)=ISR(K-1,J)
C  ADDITION OF CHANNEL NOISE - H/D QUANTIZATION - COUNTING OF CHANNEL
C ERRORS, INJECTED IN MESSAGE AND IN PARITY CHECK DIGITS.-
      DO 210 I=1,K0
      SN=G05DDF(0.0,RMSN)
      POS1=POS1+SN
      POS2=POS2+SN**2
      IRA(1,I)=0
      IF(SN.GE.0.5) IRA(1,I)=1
  210 IF(IJ.LE.NMNX) NM=NM+IRA(1,I)
      DO 220 I=1,IS
      SN=G05DDF(0.0,RMSN)
      POS1=POS1+SN
      POS2=POS2+SN**2
      JRA(I)=0
      IF(SN.GE.0.5) JRA(I)=1
  220 IF(IJ.LE.NMNX) MP=MP+JRA(I)
C  SYNDROME CALCULATION.-
      DO 240 J=1,IS
      ISR(1,J)=JRA(J)
      DO 240 I=1,K0
  240 ISR(1,J)=IABS(ISR(1,J)-IRA(I,JAP(J,I)))
C  ERROR SEQUENCE ESTIMATION - DECODING - SUNDROME RESETTING - COUNTING
C OF UNCORRECTED ERRORS.-
      IF(IJ.LT.K0) GO TO 140
      DO 280 J=1,K0
      ICE=IRA(K0,J)+1
      IPCS=ISR(KAR(1,J),1)
      DO 260 I=2,IS
  260 IPCS=IPCS+ISR(KAR(I,J),I)
      JPCS=IPCS/JTH
      IF(JPCS.EQ.0) GO TO 280
      NW(ICE)=NW(ICE)+1
      DO 270 I=1,IS
      KR=KAR(I,J)
  270 ISR(KR,I)=1-ISR(KR,I)
  280 AF=AF+IABS(ICE-1-JPCS)
C   CALCULATION OF AUTOCORRELATION SUMS.-
      NOB=NOB+1
      IF(NOB.LT.IEQ) GO TO 140
      JCDS=JCDS+1
      MR=MIN0(JCDS,NAP)
      MR1=MIN0(JCDS,NAP-1)
      ADE(1)=AF
      IF(AF.EQ.0) GO TO 277
      DO 275 I=1,MR
  275 AFD(I)=AFD(I)+AF*ADE(I)
  277 MR12=MR1+2
      DO 285 I=1,MR1
      J=MR12-I
  285 ADE(J)=ADE(J-1)
```

```
      MOB=0
      AF=0
C  END OF MAIN LOOP.-
  140 IF(T.LT.NMAX) GO TO 145
```

Figure A8.4.1: The *main loop* of the simulation-programme.

bits. This was used in a number of different programmes that were designed to produce various results (for example, probability of decoding error, autocorrelation function of the decoder-output error-sequence, decoding-errors per coset, error propagation, etc). All these programmes differ in their details and in the way they process the collected statistical data. The decoder uses a straightforward approach and data provided by subroutine CODAR2.

A8.4.2. A Complete Simulation Programme for Long Codes

Main programme *IKOSI5* implements the storage-saving technique, described in Example A8.3.1 (p. 536). This particular version was run in a CDC-7600 mainframe, for which the word-length was 60 (see command "LB=60", in p. 544). The arrays have been dimensioned for a particular code (with k=40). The programme needs only to read the following data: k, J, the number of blocks to be decoded (NMAX), the minimum number of error bits to be generated (MNER), the number channel-error rates to be tested* (NQE), the feedback mode (KFIB)**, the initial setting of the random-number generator (KG), instructions about the printing (or not) of any of the IA, EA & SA (MM), the syndrome threshold to be used ($T = \lceil J/2 \rceil + IDT$) and the NQE channel-error rates.

The above simulation programme is therefore very flexible. One run can produce a set of points of the net coding-gain versus the SNR / information-bit graph, as well as information about the error-rate performance of each of the k information bits of the code (indicating thus the potential for unequal error-protection). Also, the above-mentioned data can be obtained for various syndrome thresholds, so that one can determine the optimum threshold for various channel error rates (see theory on the optimum threshold, in Chapter 6). Finally, each channel error rate can be tested

* For each of them, at least NMAX blocks are considered.
** Any combination of DD, FD & 'genie' decoding can be employed.

```
      PROGRAM IKOSI5(INPUT,OUTPUT,TAPE1=INPUT,TAPE2=OUTPUT)
C  DIMENSION OF JR,JS,LUC,JET,IFB,MAA,MAB,MAC,KR & ISR IS (M-1) -
C  DIMENSION OF IPA IS (KX)X(M-1), WHERE KX = INT[(M-1.5)/LB]+1, AND
C  LB = NUMBER OF BITS/WORD - TOTAL STORAGE REQUIRED IS (M-1)*(KX+11).-
C  IS MUST BE IS<LB+1 - IF IS>LB, IS = 2, BY DEFAULT.-
C  NMAX = NUMBER OF BLOCKS TO BE DECODED.-
C  MNER = MINIMUM NUMBER OF CHANNEL ERRORS, IN MESSAGE DIGITS, IN EACH
C  RUN - NMAX WILL BE ADJUSTED IF IT RESULTS IN LESS ERRORS THAN MNER.-
C  NOE = NUMBER OF RUNS.-
C  KFIB CONTROLS THE SYNDROME RESET MODE: IF MD1,MD2 & MD3 DENOTE THE
C  'TRANSMITTER F/B', 'NO F/B' & 'DECODER O/P F/B' MODE RESPECTIVELY,
C  THEN, IF KFIB = 1,2,3,4,5,6,7  EACH RUN IS COMPOSED BY MD1,MD2,MD3,
C  MD1 & MD2, MD1 & MD3, MD2 & MD3, MD1 & MD2 & MD3 ,  RESPECTIVELY -
C  BY DEFAULT KFIB = 3.-
C  KG = INITIAL STATE OF RANDOM NUMBER GENERATOR.-
C  MM CONTROLS THE PRINTOUT OF ARRAYS
C   1) IF MM = 0  NO ARRAYS ARE PRINTED.
C   2) IF MM = 15  CODAR3 ARRAYS ARE PRINTED.
C   3) IF 0<MM<8  MODE MM, CODAR1 ARRAY(S), ARE PRINTED.
C   4) IF 7<MM<15  MODE MM-7 CODAR1 AND CODAR3 ARRAYS ARE PRINTED.
C  DIMENSION OF PREE IS M-1 - OF PREE, PREF & PDTX IS NOC[=(M-1)/IS].-
C  IF IPU = 1(2) ONLY THE ERROR PROBABILITY FOR EACH BIT(COSET) WILL BE
C  PRINTED - IF IPU = 0, BOTH WILL BE PRINTED.-
C  IDT = DEVIATION FROM NOMINAL SYNDROME THRESHOLD - IF ITH IS OUTSIDE
C  [0,IS], IDT = 0.-
      DIMENSION JR(40),JS(40),LUC(40),JET(40),IFB(40)
      DIMENSION MAA(40),MAB(40),MAC(40),KR(40),ISR(40)
      DIMENSION IRA(1,40)
      DIMENSION PREE(40),PREE(20),PREF(20),PDTX(20)
      DIMENSION NV(2),CG(63),CEP(63),GEPN(63),GSPN(63),NCCHE(63),CEF(63)
     1,NEME(63),PDPE(63,3),PDFCR(63,3),PDFER(63,3),ISC(61),ISE(61)
      COMMON/COD1/IA,IB/MAIN/LB
C  DATA.-
      CALL SECOND(T1)
      READ(1,500) M,IS,NMAX,MNER,NOE,KFIB,KG
      READ(1,500) MM,IPU,IDT
      READ(1,510) (GO(I),I=1,NOE)
      LB=60
C   CHECK FOR EXISTENCE OF THE CODE - CALCULATION OF ARRAYS.-
      IF(IS.GT.LB) IS=2
      P=0.5
      K0=M-1
      IA=-1
      IB=(-1)*K0
      KR(1)=0
      IF(MM.GE.8) KR(1)=1
      CALL CODAR3(K0,IS,JR,JS,LUC,JET,IFB,MAA,MAB,MAC,KR)
      IF(KR(1).GT.0) GO TO 110
      WRITE(2,520) M,IS
      STOP
C   CALCULATION OF CODE PARAMETERS.-
  110 N0=K0+IS
      NA=N0*K0
      NE=NEFEL1(M,IS)
      R1=FLOAT(NA)/NE
      IT=IS/2
      R2=FLOAT(IT)/NE
```

```
      IX=MOD(KL,2)
      IR1=K(*((IS-1)*IX+1)/IS
      IR2=IR1+1+IX
      IRC=FLOAT(NA)/IT+0.5
      RCPC=100.0*IT/NA
      MM=AMOD(FLOAT(MM),7.5)+0.5
      IF(MM.NE.0) CALL CODARI(M,IS,MM)
      KX=(K0-0.5)/LB+1
      KX0=K0/LB
      KXR=K0-LB*KX0
      MAS1=MASK(1)
      MAS2=SHIFT(1,LB-KYR-1)
      MAS3=SHIFT(1,IS-1)
      MAS4=COMPL(MASK(LB-IS+1))
      INE=LB+1
      INC=IS-INE
      NOC=K /IS
      INXS=INXP=1
      IF(NOC.LE.8.OR.NOC.GT.18.AND.NOC.LE.23) INXS=2
      IF(NOC.GT.18) INXP=2
      ISS=-1-IS
      R=FLOAT(K0)/NC
      NMXX=MNER*N /(IS/2)
      ITH=(IS+1)/2
      IF(ABS(IDT+ITH-(IS-1.0)/2).GT.(IS-1.0)/2) IDT=0
      ITH=ITH+IDT
      IS1=IS+1
      JTH=ITH+1
      IF(KFIB.LT.1.OR.KFIB.GT.7) KFIB=3
      CALL DATE(A)
      PEMAX=CHAMER(0.5,R)*100
      QEMAX=PEMAX/RCPC
      LFI1=LFI2=LFI3=0
      NFIB=(KFIB-0.5)/3+1
      CALL SECOND(T2)
      TT1=(T2-T1)/NFIB/NQE
C  OUTER LOOP.-
      DO 100 JW=1,NQE
C   CALCULATION OF SIMULATION PARAMETERS.-
      CALL SECOND(T1)
      QE=QG(JW)
      PE=QE*INT(IS/2.0)/NA
      SNT=SIMORAC(PE)
      RMSN=1/SNT
      NMNX=MAX1(FLOAT(NMAX),NMXX/QE+0.5)
      MMAX=NMNX+M-2
      NN=NMNY*K0
      MM=NN+IS*NMNX
      NMNS=NMNX*IS
C   F/B MODE CONTROL LOOP.-
      IND=KFIB
      CALL SECOND(T2)
      TT2=(T2-T1)/NFIB
  105 GO TO(112,114,116,112,112,114,112,100,100,100,114,116,116,114,100,
     1100,100,100,100,100,116,100,100,100,100,100,100,100),IND
  112 WRITE(2,530) A,IP1,IR2,N0,K0
      IFIB=-1
```

```
      LFI1=1
      GO TO 118
  114 WRITE(2,540) A,IR1,IR2,N.,K.
      IFIB=0
      LFI2=1
      GO TO 118
  116 WRITE(2,550) A,IR1,IR2,NS,K0
      LFI3=IFIB=1
  118 JFIB=IABS(IFIB)
      CALL SECOND(T1)
      IF1=JFIB*(IFIB+1)/2
      IF2=JFIB*(IFIB-1)/2
      IND=IND+7
      LFIB=IFIB+2
      WRITE(2,580) NMNX,NN,MM
C  INITIALIZATIONS.-
      NE=NM=NW(1)=NW(2)=0
      POS1=POS2=0.0
      CALL GGSCBF(KG)
      DO 125 I=1,IS1
  125 ISC(I)=ISE(I)=0
      DO 120 I=1,K0
      PRED(I)=0
      ISR(I)=0
      DO 120 J=1,KX
  120 IRA(J,I)=0
C  MAIN LOOP.-
      IJ=0
  145 IJ=IJ+1
C  SHIFT.-
      DO 130 I=2,K0
      K=K0+2-I
      ISR(K)=ISR(K-1)
      DO 130 J=1,KX
  130 IRA(J,K)=IRA(J,K-1)
C  ADDITION OF CHANNEL NOISE - A/D QUANTIZATION - COUNTING OF CHANNEL
C ERRORS, INJECTED IN MESSAGE AND IN PARITY CHECK DIGITS.-
      IF(KX..EQ.0) GO TO 150
      DO 160 I=1,KX1
      INRA=0
      DO 170 J=1,LE
      SN=GGNDDF(0.0,PMSN)
      POS1=POS1+SN
      POS2=POS2+SN**2
      IF(SN.LE.0.5) GO TO 170
      IF(IJ.LE.NMNX) NM=NM+1
      INRA=OR(INRA,MAS1)
  170 INRA=SHIFT(INRA,1)
  160 IPA(I,1)=INRA
      IF(KXR.LQ.0) GO TO 180
  150 INRA=0
      DO 190 I=1,KXR
      SN=GGNDDF(0.0,RMSN)
      POS1=POS1+SN
      POS2=POS2+SN**2
      IF(SN.LE.0.5) GO TO 190
      IF(IJ.LE.NMNX) NP=NP+1
```

```
      INRA=OR(INRA,MAS2)
  190 INRA=SHIFT(INRA,1)
      IRA(KX,1)=INRA
  180 JRA=0
      DO 200 I=1,IS
      SN=GG5DDF(0.0,RMSN)
      POS1=POS1+SN
      POS2=POS2+SN**2
      JRA=SHIFT(JRA,1)
      IF(SN.LE.0.5) GO TO 200
      IF(IJ.LE.NMNY) NP=NP+1
      JRA=OR(JRA,1)
  200 CONTINUE
C  SYNDROME CALCULATION.-
      DO 210 I=1,K
      KRA=SHIFT(IRA(JS(I),I),JR(I)+INC)
      KRA=AND(KRA,MAA(I))
      GO TO(210,220,230,240,250,260),JET(I)
  220 LRA=SHIFT(IRA(JS(I),I),IFE(I))
      LRA=AND(LRA,MAR(I))
      KRA=OR(KRA,LRA)
      GO TO 210
  230 LRA=SHIFT(IRA(JS(I)+1,I),IFE(I))
      LRA=AND(LRA,MAR(I))
      KRA=OR(KRA,LRA)
      GO TO 210
  240 LRA=AND(IRA(JS(I)-1,I),MAR(I))
      KRA=OR(KRA,LRA)
      GO TO 210
  250 LRA=SHIFT(IRA(JS(I)+1,I),IFE(I))
      LRA=AND(LRA,MAR(I))
      MRA=SHIFT(IRA(JS(I),I),JR(I)-INE)
      MRA=AND(MRA,MAC(I))
      KRA=OR(KRA,LRA,MRA)
      GO TO 210
  260 LRA=SHIFT(IRA(JS(I)-1,I),IFE(I))
      LRA=AND(LRA,MAR(I))
      MRA=SHIFT(IRA(JS(I),I),JR(I)-INE)
      MRA=AND(MRA,MAC(I))
      KRA=OR(KRA,LRA,MRA)
  210 JRA=XOR(JRA,KRA)
      ISR(I)=JRA
C  ERROR SEQUENCE ESTIMATION - DECODING - SUNDROME RESETTING - COUNTING
C  OF UNCORRECTED ERRORS.-
      IF(IJ.LT.KS) GO TO 140
      KLB=LB
      KXI=1
      IDE=IRA(1,KS)
      IST=ISS+1
      DO 280 I=1,NOC
      IST=IST+IS
      ISFB=0
      DO 270 J=1,IS
      ISH=ISS+J
      KN=IST
      IFCS=0
      DO 275 K=1,IS
```

```
      KK=KK+1
      IF(ISH.EQ.0) ISF=ISS+1
      ISH=ISH+1
      JSR=SHIFT(ISR(KP(KK)),ISH)
      JSR=AND(JSR,1)
  275 IPCS=IPCS+JSR
      JPCS=IPCS/JTP
      IDE=SHIFT(IDE,1)
      ICE=1+AND(1,IDE)
      JIST=J+IST
      JDE=IABS(ICE-1-JPCS)
      PRED(JIST)=PRED(JIST)+JDE
      IPCS1=IPCS+1
      ISC(IPCS1)=ISC(IPCS1)+1-JDE
      ISE(IPCS1)=ISE(IPCS1)+JDE
      NW(ICE)=NW(ICE)+JPCS
      IF(JIST.LE.KLE) GO TO 287
      KLB=KLB+LB
      KXI=KXI+1
      IDE=IPA(KXI,KL)
  287 IF(IF2*(ICE-1).EQ.IF1*JPCS) GO TO 270
      IKFB=SHIFT(1,IS-J)
      ISFB=OR(ISFB,IKFB)
  270 CONTINUE
      IF(ISFB.EQ. ) GO TO 280
      KK=IST
      DO 285 J=1,IS
      KK=KK+1
      KRK=KR(KK)
      ISR(KRK)=XOR(ISR(KRK),ISFB)
      IH=AND(MAS4,SHIFT(ISFB,-1))
      IG=AND(MAS3,SHIFT(ISFB,IS-1))
  285 ISFB=OR(IH,IG)
  280 CONTINUE
C     END OF MAIN LOOP.-
  14  IF(IJ.LT.MMAX) GO TO 145
C     OUTPUT OF GENERAL SIMULATION RESULTS.-
      ER=10.*PE
      NCE=NP+NM
      MN=MP-NN
      NW(2)=NN-NW(2)
      NDE=NW(1)+NW(2)
      NNCH=NN-NM
      PCE=CER(JW)=100.0*NCE/MM
      GLM=PCE/RCPC
      WRITE(2,584) ITM,IDT
      WRITE(2,585) GEN,GE
      PCEP=100.0*NP/MN
      PCEM=100.0*NM/NN
      PLE=PODE(JW,LFIB)=100.0*NDE/MN
      PLEP=PDEER(JW,LFIB)=100.0*NW(2)/NM
      PDEA=PDECR(JW,LFIB)=100.0*NW(1)/NNCH
      BSC=1-EDROPY(PCE/100)
      CFF(JW)=R/BSC*100
      WRITE(2,589) BSC,CEF(JW)
      NOCHE(JW)=NCE
      NDMB(JW)=NMMX
```

```
      FEC=PCE/10.
      IF(IFIB.GE.5) GO TO 292
      TPD=QEM
      ISR(1)=1
      ISR(2)=NOC
      CALL PRODE1(KC,NCC,TPD,IDT,PCTX,ISR)
      EERT=TPD
      GO TO 293
  292 IF(IFIB.EQ.1) GO TO 295
      EERT=PRODE2(KC,IS,QEM,ITH)
      DO 294 I=1,NCC
  294 PCTY(I)=EERT
  293 PDET=EERT*FCE
  295 WRITE(2,590) NCE,MM,PCE,ER,IP,MI,PCEP,ER,NM,KN,PCEM,ER,NDE,KN,PDE
      IF(IFIB.NE.1) WRITE(2,591) PDET
      WRITE(2,594) NW(2),NM,PDEP
      IF(PDEA.EQ.0. ) GO TO 300
      RPA=PDEP/PDEA
      WRITE(2,595) RPA
      GO TO 310
  300 WRITE(2,596)
  310 WRITE(2,597) NW(1),MNCH,PDEA
      FED=PDE/100
      DLM=POS1/MAX/KD
      PCM=POS2/MAX/KD
      PCMT=SMSL**2
      SNREF=SINORA (FEC)
      SNR=1/SQRT(FCM)
      WRITE(2,599) DLM,FCM,FCMT,SNR,SNREF,SNT
      SNRDB=20*ALOG10(SNR)
      SA=SNR**2/8

      SAD=SNRDB-10*ALOG10(8.0)

      SAN=SA/R

      BED=(-10)*ALOG10(R)
      SAND=SAD+BED

      BE=1/R
      EFR=PED/PEC
      EU=S15ACF(SNR/2/SQRT(R*2),D)
      EUP=EU*100
      IF(IFIB.EQ.1) GO TO 315
      GERT=EUP/PDET
      SNRUT=SINORAC(PDET/100)
      GSPDT=20*ALOG10(SNRUT)-10*ALOG10(4.0)-SAND
  315 WRITE(2,600) SNR,SA,SAD,        SAN,SAND,        R,BE,BED,PED,
     1PEC,EFR
      IF(IFIB.NE.1) WRITE(2,601) EERT
      WRITE(2,605) EU,EU,EUP,EU,PED
      IF(IFIB.NE.1) WRITE(2,606) GERT
      IF(PED.EQ.0) GO TO 291
      GER=EU/PED
      SNRU=SINORA (PED)
      SNRUD=20*ALOG10(SNRU)
      SUA=SNRU**2/
```

```
      SUAD=SNRUD-10*ALOG10(8.0)

      GSP=SUA/SAN
      GSPD=SUAD-SAND
      IF(LFIB.EQ.3) GERN(JW)=GER
      IF(LFIB.LQ.3) GSPN(JW)=GSPD
      WRITE(2,610) GER,GER,SNRU,SNRUD,SNRU,SUA,SUAD,               SUA,
     1SAN,GSP,GSPD
      IF(IFIB.NE.1) WRITE(2,611) GSPDT
C     CALCULATION & OUTPUT OF SYNDROME VOTE DISTRIBUTION.-
      IF(IFIB.EQ.-1) WRITE(2,530) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.0) WRITE(2,540) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.1) WRITE(2,550) A,IR1,IR2,N0,K0
      WRITE(2,700) IS,GEM
      DO 390 I=1,IS1
      LS=I-1
      PSE=ISE(I)*100.0/NDE
      LCD=NN-NDE
      PSC=ISC(I)*100.0/LCD
      IST=ISE(I)+ISC(I)
      PST=IST*100.0/NN
  390 WRITE(2,750) LS,JTH,ISE(I),NDE,PSE,ISC(I),LCD,PSC,IST,NN,PST
      IF(IPU.EQ.2) GO TO 400
C     CALCULATION & OUTPUT OF DIGIT DISTRIBUTION OF DECODING ERRORS.-
      ISR(1)=-1
      CALL ORDER(K0,PRED,ISR)
      IF(IFIB.EQ.-1) WRITE(2,530) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.0) WRITE(2,540) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.1) WRITE(2,550) A,IR1,IR2,N0,K0
      WRITE(2,700) GEM
      DO 380 I=1,K0
      IPRED=PRED(I)
      DERP=100.0*IPRED/NMNX
      DD=(DERP/PDE-1)*100
      J=ISR(I)
      JPRED=PRED(J)
      DJRP=100.0*JPRED/NMNX
      DJ=(DJRP/PDE-1)*100
      IF(MOD(I,IS).EQ.1) WRITE(2,720)
  380 WRITE(2,710) IPRED,NMNX,DERP,I,DD,JPRED,NMNX,DJRP,J,DJ
  400 IF(IPU.EQ.1) GO TO 299
C     CALCULATION & OUTPUT OF COSET DISTRIBUTION OF DECODING ERRORS.-
      IF(IFIB.EQ.-1) WRITE(2,530) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.0) WRITE(2,540) A,IR1,IR2,N0,K0
      IF(IFIB.EQ.1) WRITE(2,550) A,IR1,IR2,N0,K0
      WRITE(2,730) GEM
      IF(IFIB.EQ.1) WRITE(2,732)
      IF(IFIB.LE.0) WRITE(2,734)
      KC=0
      DO 405 I=1,NOC
      PRED(I)=PRED(KC+1)
      DO 410 J=2,IS
  410 PRED(I)=PRED(I)+PRED(KC+J)
  405 KC=KC+IS
      ISR(1)=-1
      CALL ORDER(NOC,PRED,ISR)
```

```
      DO 415 I=1,NOC
      IPRED=PRED(I)
      DERP=100.0*IPRED/NMNS
      DD=(DERP/PDE-1)*100
      J=ISR(I)
      JPRED=PRED(J)
      DJRP=100.0*JPRED/NMNS
      DJ=(DJRP/PDE-1)*100
      IF(IFIB.EQ.1) GO TO 417
      DERT=PDTX(I)*PCE
      DDT=(DERT/PDET-1)*100
      DJRT=PDTX(J)*PCE
      DJT=(DJRT/PDET-1)*100
      WRITE(2,715) IPRED,NMNS,DERP,DERT,I,DD,DDT,JPRED,NMNS,DJRP,DJRT,J,
     1DJ,DJT
      GO TO 415
  417 WRITE(2,710) IPRED,NMNS,DERP,I,DD,JPRED,NMNS,DJRP,J,DJ
  415 CONTINUE
C   CALCULATION OF DECODER ERRORS DUE TO DECODER OR NO F/B.-
      IF(KFIB.LE.4) GO TO 299
      IF(IFIB.GE.0) GO TO 420
      DO 425 I=1,NOC
  425 PREF(I)=PRED(I)
      GO TO 299
  420 IF(IFIB.EQ.1) GO TO 430
      DO 435 I=1,NOC
  435 PREF(I)=PRED(I)
      GO TO 299
  430 WRITE(2,500) A,IR1,IR2,NO,KC
      IF(KFIB.EQ.6) GO TO 440
C   CALCULATION & OUTPUT OF DECODER ERRORS DUE TO DECODER F/B.-
      DO 445 I=1,NOC
  445 PREE(I)=PRED(I)-PREF(I)
      ISR(1)=-1
      CALL ORDER(NOC,PREE,ISR)
      WRITE(2,740) GEM
      DO 450 I=1,NOC
      IPD=PRED(I)
      IPE=PREE(I)
      J=ISR(I)
      JPD=PRED(J)
      JPE=PREE(J)
      IF(INXS.EQ.2) WRITE(2,770)
      IPT=IPD-IPE
      DPI=DPJ=99999.999
      IF(IPT.NE.0) DPI=100.0*IPE/IPT
      JPT=JPD-JPE
      IF(JPT.NE.0) DPJ=100.0*JPE/JPT
  450 WRITE(2,760) I,IPD,IPT,IPT,IPE,IPT,DPI,J,JPE,JPT,DPJ
      IF(KFIB.EQ.5) GO TO 299
C   CALCULATION & OUTPUT OF DECODER ERRORS DUE TO NO F/B.-
  440 DO 455 I=1,NOC
  455 PREF(I)=PREF(I)-PRED(I)
      ISR(1)=-1
      CALL ORDER(NOC,PREF,ISR)
      IF(INXP.EQ.1) WRITE(2,720)
      IF(INXP.EQ.2) WRITE(2,500) A,IR1,IR2,NO,KC
```

```
      WRITE(2,750) GEP
      DO 460 I=1,NCC
      IPD=PRED(I)
      IPF=PREF(I)
      J=ISR(I)
      JPD=PRED(J)
      JPF=PREF(J)
      IF(IMXS.EQ.2) WRITE(2,770)
      IPN=IPD+IPF
      JPN=JPD+JPF
      DPI=DPJ=99999.999
      IF(IPD.NE.0) DPI=100.0*IPF/IPD
      IF(JPD.NE.0) DPJ=100.0*JPF/JPD
  460 WRITE(2,760) I,IPN,IPD,IPD,IPF,IPD,DPI,J,JPF,JPD,DPJ
      GO TO 299
  290 WRITE(2,620)
      IF(IFIL.NE.1) WRITE(2,611) GSPDT
      IF(LFIB.EQ.3) GERN(JW)=GSPN(JW)=9999.999
  299 CALL SECOND(T2)
      TT=T2-T1+TT1+TT2
      WRITE(2,614)
      WRITE(2,615) TT
      GO TO 105
  100 CONTINUE
      WRITE(2,560) A,IR1,IR2,N ,K
      WRITE(2,570) NJ,K2,KJ,IS,IR1,IR2,NA,NE,R1,IT,NE,R2,IT,NA,IRC,RCPC,
     1PEMAX,GEMAX,KG
      WRITE(2,560) A,IR1,IR2,NC,KC
      IF(LFI3.EQ. ) GO TO 320
      WRITE(2,630)
      DO 330 I=1,NOE
      EER=PODE(I,3)/CER(I)
      QGE=CER(I)/RCPC
  330 WRITE(2,640) CER(I),QGE,PODE(I,3),EER,GERN(I),GSPN(I),NOCHE(I),
     1CEF(I),NDME(I)
  320 WRITE(2,560) A,IR1,IR2,NJ,KC
      WRITE(2,650)
      DO 340 I=1,NOE
      QGE=CER(I)/RCPC
      WRITE(2,660) CER(I),QGE,CEF(I)
      IF(LFI1.EQ.0) GO TO 350
      RAT=999.999
      IF(PDECR(I,1).NE.0) RAT=PDEER(I,1)/PDECR(I,1)
      IF(PDECR(I,1)+PDEER(I,1).EQ.0) RAT=-1
      WRITE(2,670) PODE(I,1),PDECR(I,1),PDEER(I,1),RAT
  350 IF(LFI2.EQ.0) GO TO 360
      RAT=999.999
      IF(PDECR(I,2).NE.0) RAT=PDEER(I,2)/PDECR(I,2)
      IF(PDECR(I,2)+PDEER(I,2).EQ.0) RAT=-1
      WRITE(2,690) PODE(I,2),PDECR(I,2),PDEER(I,2),RAT
  360 IF(LFI3.EQ. ) GO TO 340
      RAT=999.999
      IF(PDECR(I,3).NE.0) RAT=PDEER(I,3)/PDECR(I,3)
      IF(PDECR(I,3)+PDEER(I,3).EQ.0) RAT=-1
      WRITE(2,680) PODE(I,3),PDECR(I,3),PDEER(I,3),RAT
  340 CONTINUE
      CALL SECOND(T2)
```

```
      WRITE(2,610) T2
      STOP
  500 FORMAT(7I10)
  510 FORMAT(9(7F10.2/))
  520 FORMAT(1H1,5X,10(5X/)/1H0,10X, 'THERE EXISTS NO CODE OF MODULO M
     1=',I4,' AND ORDER IS =',I4)
  530 FORMAT(1H1,3X,'-----> SYNDROME REGISTER IS RESET FROM TRANSMITTER
     1<',6('-'),'> IKOSI5 <-->',A10,'<--->',I4,'/',I3,' <--> (',I4,',',I
     34,') <-----')
  540 FORMAT(1H1,3X,'-----> SYNDROME REGISTER IS NOT RESET <',12('-'),'>
     1 IKOSI5 <--->',A10,'<',6('-'),'>',I4,'/',I3,' <---> (',I4,',',I4,'
     2) <-----')
  550 FORMAT(1H1,3X,'-----> SYNDROME REGISTER IS RESET FROM DECODER O/P
     1<',6('-'),'> IKOSI5 <-->',A10,'<--->',I4,'/',I3,' <--> (',I4,',',I
     24,') <-----')
  560 FORMAT(1H1,21X,'-----> IKOSI5 <----->',A10,'<---------->',I4,'/',I
     13,' <-----> (',I4,',',I4,') <-----')
  570 FORMAT(1H0,5X,'CODE LENGTH =',I4/1H0,5X,'NUMBER OF MESSAGE DIGITS
     1=',I4/1H0, 5X,     'BLOCK-CONSTRAINT LENGTH =',I4/1H0,5X,'NUMBER
     2OF ORTHOGONAL CHECK-SUMS =',I4/1H0,5X,'CODE RATE =',I3,'/',I3/1H0,
     3 5X,'ACTUAL CONSTRAINT LENGTH =',I6/1H0,5X,'EFFECTIVE CONSTRAINT L
     4ENGTH =',I5/1H0,5X,'ACTUAL C.L./EFFECTIVE C.L. = NA/NE =',F7.2/1H0
     5, 5X,'RELATIVE ERROR CORRECTING CAPABILITY = [J/2]/NE =',I4,'/',I5
     6,' =',F8.5/1H0,5X,   'GUARANTEED ERROR CORRECTING CAPABILITY =',
     7I3, ' M/D ERRORS IN ANY', I6, ' DIGITS = 1 M/D ERROR /',   I6,  '
     8DIGITS =',F9.5,'%'///1H0,5X,'MAX BSC ERROR RATE =',F8.4,'%'/1H0,5X
     9,'MAXIMUM QE =',F6.2//1H0,5X,'INITIAL STATE OF RANDOM NUMBER GENER
     AATOR = KG =',I3)
  58  FORMAT(1H0,5X,  'NUMBER OF DECODED MESSAGE BLOCKS =',I6/1H0,    5X
     1,   'NUMBER OF DECODED MESSAGE DIGITS =',    I8,32X,   'NUMBER OF
     2INJECTED NOISE DIGITS =',I8)
  584 FORMAT(1H0,3X,'-----> SYNDROME THRESHOLD =',I3,'   (DEVIATION =',I
     13,')')
  585 FORMAT(1H0,3X,'-----> QE = CHANNEL ERROR RATE/GUARANTEED ERROR COR
     1RECTING CAPABILITY =',F6.2,'   [',F6.2,']')
  589 FORMAT(1H0,3X,'-----> BSC CAPACITY =',F9.7,' BITS/SYMBOL',10X,
     1'-----> INFORMATION TRANSMISSION RATE IS',F7.2,'% OF CHANNEL CAPAC
     2ITY')
  590 FORMAT(1H0,5X,'CHANNEL ERROR RATE =',I8,'/',I8,' =',F8.3,'%   [',F
     16.3,'%]'/1H0,5X,  'PROBABILITY OF A CHANNEL ERROR IN A PARITY-CHEC
     2K DIGIT =',   I6, '/', I8,' =', F8.3,'%   [',F6.3,  '%]'/1H0,5X,
     3'PROBABILITY OF A CHANNEL ERROR IN A MESSAGE DIGIT =',I8,'/',I8,
     4' =', F8.3,  '%   [',F6.3,'%]'//1H0,5X,    'PROBABILITY OF A DECOD
     5ER ERROR =', I8, '/',I8,  ' =', F9.4, '%')
  591 FORMAT(1H+,69X,'[',F9.4,'%]')
  594 FORMAT(1H0,5X,'PROBABILITY OF ERRONEOUSLY DECODING AN ERRONEOUSLY
     1RECEIVED DIGIT =',I8,'/',I8,'=',F9.4,'%'/1H , 102X, 'RATIO =')
  595 FORMAT(1H+,109X,F8.2)
  596 FORMAT(1H+,111X, '?')
  597 FORMAT(1H ,5X,'PROBABILITY OF ERRONEOUSLY DECODING A CORRECTLY REC
     1EIVED DIGIT =',I8,'/',I8,' =',F9.4,'%'/)
  599 FORMAT(1H0,5X,'NOISE D.C. LEVEL =',  F8.5,  20X,'NOISE POWER =',
     1F8.5,'  [',F7.5,']'/1H ,5X,              '1/RMS NOISE =',  F6.3,
     2'(CHANN. MEASUREM.)',  5X,'=',F6.3,    '(CORRESP. TO ACT. ERROR RAT
     3E)', 5X,'=',F6.3, '(THEOR.)'/)
  600 FORMAT(1H0,34X,'=',F7.3,'**2/4 =',F7.3,' =',F7.2,     'Db  (FOR ANT
     1IPODAL TRANSMISSION)'/1H ,    5X,             'SIGNAL-TO-NOISE      R
```

```
     4ATIO =*
     5                  //1H0,62X,*=*,F7.3,* =*,F7.2,*DB  (FOR ANTIPODAL
     6TRANSMISSION)*/1H ,3X,*-----> SIGNAL-TO-NOISE        RATIO PER INFO
     7RMATION SYMBOL =*
     8             //1H ,5X,*BANDWIDTH EXPANSION = 1/*,F6.4,2(* =*,F6.3),
     9*DB = INCREASE IN NOISE POWER*/1H0, 3X, *-----> ERROR EXTENSION RA
     ATIO =*,1PE9.2,*/*,E9.2,* =*,0PF8.4)
 601 FORMAT(1H+,65X,*[*,F8.4,*]*)
 605 FORMAT(1H0,5X,*ERROR RATE FOR UNCODED TRANSMISSION AND EQUAL SNR/I
    1INFORMATION SYMBOL =*,1PE9.2,* =*,0PF10.7,* =*,F10.5,*%*/1H0,3X,*--
    2---> NORMALIZED GAIN IN ERROR RATE =*,1PE9.2,*/*,E9.2,* =*)
 606 FORMAT(1H+,88X,*[*,F9.4,*]*)
 610 FORMAT(1H+, 64X, 1PE10.3, * =*,0PF9.4/1H0, 5X,
    1    *1/R.M.S. NOISE RATIO FOR UNCODED TRANSMISSION & EQUAL ERROR RA
    2TE =*, F7.3, * =*,F6.2, *DB*/1H , 5X,*SIGNAL-TO-NOISE        RATIO
    3FOR UNCODED ANTIPODAL TRANSMISSION & EQUAL ERROR RATE =*,F7.3,
    4***2/4 =*,F7.3,  * =*,F6.2, *DB*
    5
    6                                        /1H0, 3X, *-----> NORMALIZED GA
    7IN IN SIGNAL POWER =*, F7.3, */*,F7.3, * =*, F7.3, * =*,F9.4,*DB*)
 611 FORMAT(1H+,83X,*[*,F9.4,*DB]*)
 614 FORMAT(/1H0,6(*<*,10(*-*),*>*,6X),*<*,10(*-*),*>*)
 615 FORMAT(1H0,100X,*(P.T. =*,F7.2,* SECS)*)
 616 FORMAT(///1H0,62X,*[TOTAL PROCESSING TIME =*,F8.2,* SECS]*)
 620 FORMAT(1H+,65X, *ARBITRARILY LARGE*/1H0,5X,*PEAK-TO-PEAK SIGNAL TO
    1 R.M.S. NOISE RATIO FOR UNCODED TRANSMISSION & EQUAL ERROR RATE =
    2ARBITRARILY LARGE*/1H0,3X,*-----> NORMALIZED GAIN IN SIGNAL POWER
    3= ARBITRARILY LARGE*///)
 630 FORMAT(1H0,*CER(%)      QE      PDE(%)      EER        NGER      NGSP(D
    1B)      NOCHE     CEF(%)     NDMB*)
 640 FORMAT(1H ,F6.3,F8.2,F10.3,F9.3,2F11.3,I11,F10.3,I9)
 650 FORMAT(1H0,*CER(%)*,10X,*PDE(%)*,16X,*PDE/CR(%)*,14X,*PDE/ER(%)*,1
    11X,*PDE RATIO (EF/CR)*,7X,*QE      CEF(%)*/1H ,9X,3(*   TX      DE
    2  NC      *),*  TX       DE       NC*)
 660 FORMAT(1H ,F6.3,94X,F8.2,F10.3)
 670 FORMAT(1H+,F15.3,3F23.3)
 680 FORMAT(1H+,F22.3,2F23.3,F24.3)
 690 FORMAT(1H+,6X,3F23.3,F25.3)
 700 FORMAT(1H0,18X,*PD(J) = PROBABILITY OF DECODER ERROR IN THE JTH DI
    1GIT OF A SUBLOCK      QE =*,F7.2/1H ,18X,82(*-*)/1H0,33X,*DE(J) = P
    2ERCENTAGE DEVIATION FROM UNIFORM DISTRIBUTION*/1H ,33X,54(*-*)//1H
    30,2(11X,*PD(J)    (%)*,12X,*J*,4X,*DE(J)    (%)*,4X))
 710 FORMAT(1H ,2(I10,*/*,I6,* =*,F9.4,I8,F13.3,5X))
 715 FORMAT(1H ,I5,*/*,I6,* =*,F7.4,* [*,F7.4,*]*,I6,F10.3,* [*,F8.3,*]
    1*,I9,*/*,I6,* =*,F7.4,* [*,F7.4,*]*,I6,F10.3,* [*,F8.3,*]*)
 720 FORMAT(1H ,120(*-*))
 730 FORMAT(1H0,18X,*PD(J) = PROBABILITY OF DECODER ERROR IN THE JTH CO
    1SET OF A SUBLOCK      QE =*,F7.2/1H ,18X,82(*-*)/1H0,33X,*DE(J) = P
    2ERCENTAGE DEVIATION FROM UNIFORM DISTRIBUTION*/1H ,33X,54(*-*)//)
 732 FORMAT(1H0,2(11X,*PD(J)    (%)*,12X,*J*,4X,*DE(J)    (%)*,4X))
 734 FORMAT(1H0,10X,*PD(J)    (%)*,14X,*J*,6X,*DE(J)    (%)*,17X,*PD(J)
    1 (%)*,14X,*J*,6X,*DE(J)    (%)*)
 740 FORMAT(1H0,*DPTD(J) = [ND(J)-NT(J)]/NT(J) WHERE: NT(J)[ND(J)] = NO
    1 OF DECODER ERRORS IN COSET J OF A SUBLOCK, WITH TX(DE) F/B*/1H ,1
    213(*-*)/1H0,*DPTD(J) = % IN(DE)CREASE IN THE NUMBER OF DECODER ERR
    3ORS, IN THE JTH COSET OF A SUBLOCK DUE TO (IN)CORRECT DECODER F/B*
    4/1H ,118(*-*)/1H ,54X,*QE =*,F7.2/1H ,54X,11(*-*)//1H0,11X,*J*,4X,
```

```
AM IKOSI5      76/76   OPT=1  PMDMP                         FTN 4.8+538          15

   5'[ ND(J)- NT(J)]/ NT(J) =  ND-NT/ NT(J) =  DPTD(J) %',13X,'J',5X,'
   6ND-NT/ NT(J) =  DPTD(J) %')
750 FORMAT(1H0,'DPDN(J) = [NN(J)-ND(J)]/ND(J) WHERE: NN(J)[ND(J)] = NO
   1 OF DECODER ERRORS IN COSET J OF A SUBLOCK, WITH NO(DE) F/B'/1H ,1
   213('-')/1H ,'DIFN(J) = % IN(DE)CREASE IN THE NUMBER OF DECODER ERR
   3ORS, IN THE JTH COSET OF A SUBLOCK DUE TO NO SYNDROME RESETTING'/1
   4H ,116('-')/1H ,54X,'QE =',F7.2/1H ,54X,11('-')//1H0,11X,'J',4X,'[
   5 NN(J)- ND(J)]/ ND(J) =  NN-ND/ ND(J) =  DPDN(J) %',13X,'J',5X,'NN
   6-ND/ ND(J) =  DIFN(J) %')
760 FORMAT(1H ,I12,4X,'(',I6,'-',I6,')/',I6,' =',I7,'/',I6,' =',F10.3,
   1'%',I14,4X,I6,'/',I6,' =',F10.3,'%')
770 FORMAT(1H ,1X)
780 FORMAT(1H0,26X,'PROBABILITY OF A I/(ITH+1) VOTE, BY THE',I3,' SYND
   1ROMES        GE =',F7.2/1H ,26X,68('-')///1H0,'I/(ITH+1) VOTE',11X,'F
   2OR DECODING ERROR',10X,'FOR CORRECT DECODING',23X,'TOTAL')
790 FORMAT(1H0,I6,'/',I2,I17,'/',I7,' =',F11.5,'%',2(I13,'/',I8,' =',F
   111.5,'%'))
    END
```

Figure A8.4.2: The complete FORTRAN programme for the simulation of 'long' codes, over a number of different channel error rates, and with any choice of syndrome-resetting modes and syndrome threshold; the programme also calculates results for each coset.

for any combination of syndrome-resetting modes (definite-decoding, feedback decoding & correct syndrome-resetting, or 'genie' decoding).

A processing-speed performance comparison between programme IKOSI5 and its version which used the decoder implementation of Fig. A8.4.1 concluded that the 'long'-codes version is also economical with processing-time (apart from being economical with storage space). For example, 10,000 blocks of the (144,4) code were simulated with both versions and while the older one required about 140 secs per 10,000 blocks, IKOSI5 required about 50 secs (both run on a CDC-7600 mainframe).

APPENDIX 8.5 SUBROUTINES USED BY THE MAIN PROGRAMME

In Appendix 8.1, a number of routines necessary for the implementation of the chosen code, was presented. Appendix 8.5 will introduce the subroutines that are necessary for the processing of the simulation results.

According to reln (A6.2.5), the BSC error rate, P_e, must be such that the code rate, R, satisfies $R < 1 + P_e\log_2 P_e + (1-P_e)\log_2(1-P_e)$. Function CHAMER(0.5,R)* returns the maximum allowed P_e, for a given R, by solving eqn $P_e\log_2 P_e + (1-P_e)\log_2(1-P_e) + 1 - R = 0$.**

From eqn (1.4), $P_e - \frac{1}{2}\mathrm{erfc}(\sqrt{\Gamma}) = 0$ must be solved for Γ, given P_e. Function SINORA0(P_e)* solves eqn $P_e - Q(\frac{1}{2}\sigma) = 0$, for $1/\sigma$. Note, from (A1.2.26), that $Q(x) = \frac{1}{2}\mathrm{erfc}(x/\sqrt{2})$. Also, $\Gamma = (1/\sigma)^2/8$.**

Function PRODE2(k,J,P_e,T)* returns the probability of decoding error, P_d, for the (k,J) type-C5 code over the BSC with error probability P_e, under DD and with syndrome threshold T (it uses the results of Theorem 6.8, p. 164).

Subroutine PRODE1(k,J,P_e,T,Parr,Iarr) returns the (theoretical) probability of first decoding error, under FD, for each coset (in array Parr) of the (k,J) type-C5 code. P_e is the channel error rate, T is the threshold used and Iarr(i) are the cosets to be examined (see § A8.6.2., p. 561).

The routine uses the results of Theorem 6.3 (p. 157). As a consequence, it requires the facility of another routine that returns all the combinations of t, out of N things, for the calculation of the generalized means. This routine is actually incorporated into PRODE1, for practical reasons, and it is briefly described below:

Given N 'things', denoted by 1,2,...,N, one would like all the C(N,t) distinct combinations of t, out of the N. For instance, if N=5 & t=3, the C(5,3) = 5!/3!/2! = 10 combinations are listed below, in their 'natural' order:

123 124 125 134 135 145 234 235 245 345

Note that the rightmost element changes faster than the

* See Appendix 8.6 (§ A8.6.1., p. 559).

** The NAG-Library subroutine C05ADF is used to solve, numerically, the eqn.

rest and when it reaches its maximum (5), the previous element is increased by 1, the last is set at 1 plus the previous element, etc. Let array Ca(i) /i=1,2,...,t contain the current combination [if this is 135, Ca(1)=1, Ca(2)=3 & Ca(3)=5]. Consider now a pointer, IND, indicating one of the t elements. Note that the rightmost element of the combination, Ca(t), must be Ca(t) ≤ N, while the 2nd from the right Ca(t-1) ≤ N-1, etc, Ca(t-r) ≤ N-r /r=0,1,...,t-1. If t-r = IND, then:

$$Ca(IND) \le N-t+IND \quad /IND=1,2,\ldots,t \qquad (A8.5.1)$$

When Ca(IND)=N-t+IND, it means that this element has reached its maximum value. Hence the previous element, Ca(IND-1), is considered and, provided that IND≥1, test (A8.5.1) is repeated. If it fails, IND is reduced once more by one, etc. If all subsequent tests fail, and IND becomes 0, then there is no other combination. If, for a value of IND, the test succeeds, then element Ca(IND) is increased by 1, while the rest of the elements, Ca(IND+1), Ca(IND+2),..., Ca(t), take on values Ca(IND)+1, Ca(IND)+2,..., Ca(IND)+t, respectively.

<u>Specification</u>: "Given N, t & δ integers, such that N≥1, 1≤t≤N & δ≥1 and a 1 x t array Ca, return in Ca the δth next combination of t out of the N things. On entry, Ca contains the current combination. If there is no such combination, return Ca(1)=0." The algorithm (Fig. A8.5.1) is original.

```
If [N<1 or t<1 or t>N or δ<1 or Ca(i)≤Ca(i-1) /some i], then: STOP
If else, then:   IND = t
-> Step 1:  If [Ca(IND)-IND ≠ N-t], then:  Go to step 2
            If else, then:  IND = IND-1
            If (IND > 0), then:   Repeat step 1
            If else, then:  Ca(1) = 0                      --> END
   Step 2:  Ca(IND) = Ca(IND)+1
 -> Step 3:  If (IND ≠ t), then:  IND = IND+1 & Ca(IND) = Ca(IND-1)+1
                                  Repeat step 3
             If (IND = t), then:  δ = δ-1
             If (δ > 0), then:  Repeat step 1
             If else, then:                                  > END
```

<u>Figure A8.5.1</u>: Flow-chart for next combination.

Also required by PRODE1, is a reordering subroutine OR-

DER(N,Da,Po)*. This is supplied with the number, N, of elements to be reordered and two 1 x N arrays, Da & Po. Da contains the elements to be reordered (unchanged on exit). On exit, Po contains the indices of array Da in such a way that Da(Po(i)) /i=1,2,...,N are, in ascending order if on entry Po(1)=1, or in decsending order if Po(1)=otherwise.

Reordering is in descending order. If required, it is reversed at the end. Initially, Po(i) is set to i /i=1,2,...,N. There are n $\hat{=}$ $\lfloor N/2 \rfloor$ steps. In the first one, Po(1) and Po(N) are determined. In the 2nd step Po(2) & Po(N-1), etc.

For each step i=1,2,...,n, Dmx denotes the maximum between Da(Po(i)) & Da(Po(N+1-i)) and Dmn their minimum. Similarly, Po(Nmx) is the position of Dmx within Da and Po(Nmn) the position of Dmn. Before Dmx & Dmn are compared with the elements between them, M1 [=Po(Nmx)] & M2 [=Po(Nmn)] store their positions and Inn=Inx=0 (they will be used later to indicate the type of changes on Dmn & Dmx).

For all the elements Da(Po(j)) /j = i+1, i+2,..., N-i (if there are any), Dmn, Dmx, Nmn & Nmx are modified accordingly. If a minimum is found, Inn=1; if a maximum is found, Inx=1. Then, the two Po elements are set to Po(i)=Po(Nmx) and Po(N+1-i)=Po(Nmn), unless i=Nmn in which case Po(N+1-i) = M2. If a min was found (Inn=1) between Da[Po(i)] & Da(Po(N+1-i)), then this means that Po(N+1-i) took on the value of Po(Nmn); hence the latter must take the previous value of Po(Nmn), which was stored in M2. Similarly, if a max was found. The algorithm (Fig. A8.5.2) is original.

A number of other subroutines were also used (like decimal-to-binary conversion, calculation of the binomial coefficient without overflow, etc), but because their flow-chart is straightforward, they will not be mentioned. Concerning the binomial coefficient, note that**

$$\log\binom{n}{k} = \sum_{i=1}^{n}\log i - \sum_{i=1}^{k}\log i - \sum_{i=1}^{n-k}\log i = \sum_{j=k+1}^{n}\log j - \sum_{i=1}^{n-k}\log i$$

$$\Longrightarrow \log\binom{n}{k} = \sum_{i=1}^{n-k}\log(1+k/i) \qquad \text{(A8.5.2)}$$

* See Appendix 8.6 (§ A8.6.2., p. 563).

** C(n,k) may be small, but n! may still cause overflow.

```
    Da(i) = i  for i=1,2,...,N
┌─> For i=1,2,...,N/2
│   If [Da(Po(i)) ≥ Da(Po(N+1-i))], then:  Nmx=i  &  Nmn=N+1-i
│   If [Da(Po(i)) < Da(Po(N+1-i))], then:  Nmn=i  &  Nmx=N+1-i
│   Dmn=Da(Po(Nmn))  &  Dmx=Da(Po(Nmx))
│   Inn=Inx=0  &  M1=Po(Nmx)  &  M2=Po(Nmn)
│   If (2i = N), then:   Go to step 1 ────────────────────────┐
│ ┌─> For j=i+1,i+2,...,N-i                                  │
│ ├─  If [Dmn ≤ Da(Po(j)) ≤ Dmx], then:  next j              │
│ │   If [Dmn > Da(Po(j))], then:  Dmn = Da(Po(j))           │
│ ├────────────────────────────────  Nmn=j  &  Inn=1         │
│ │   If [Dmx < Da(Po(j))], then:  Dmx = Da(Po(j))           │
│ └────────────────────────────────  Nmx=j  &  Inx=1         │
│     Step 1:  Po(i) = Po(Nmx)  <────────────────────────────┘
│              If (i = Nmn), then:  Po(N+1-i) = M2
│              If else,      then:  Po(N+1-i) = Po(Nmn)
│              If (Inn=1), then:  Po(Nmn) = M2
│              If (Inx=1), then:  Po(Nmx) = M1
└──────────    next i
    END
```

Figure A8.5.2: Flow-chart for array reordering.

APPENDIX 8.6: FORTRAN PROGRAMMES FOR APPENDIX 8.5

A8.6.1. Channel Capacity, Channel Error-Rate & Probability of Decoding Error under DD

```
FUNCTION CHAMER(P,R)
COMMON/CHA/C
EXTERNAL DH
C=R*EDROPY(P)-1
CALL C[illegible]AFF(0.  ,  .5,2E-5,1E-7,DH,X,[illegible])
CHAMER=X
RETURN
END

FUNCTION DH(X)
COMMON/CHA/C
DH=EDROPY(X)+C
RETURN
END

FUNCTION EDROPY(P)
EDROPY=0
IF(P.EQ.0.OR.P.EQ.1) RETURN
EDROPY=(-1)*(P*ALOG10(P)+(1-P)*ALOG10(1-P))/ALOG10(2.0)
RETURN
END
```

Figure A8.6.1: FORTRAN programme for *function CHAMER*.

```
      FUNCTION SINORAO(PE)
      COMMON/SINO/PEE
      EXTERNAL DR
      SINORAO=0.0
      IF(PE.GE.0.5) RETURN
      PEE=PE
      A=1.705*((-1)*ALOG10(PE))**0.571-0.5
      B=A+1
  170 IF(DR(A)*DR(B).LT.0) GO TO 210
      A=A-0.5
      B=B+0.5
      A=AMAX1(A,1E-9)
      GO TO 170
  210 CALL C05ACF(A,B,1E-5,1E-9,DR,S,0)
      SINORAO=2*S
      RETURN
      END

      FUNCTION DR(X)
      COMMON/SINO/PEE
      DR=S15ACF(X,0)-PEE
      RETURN
      END
```

Figure A8.6.2: FORTRAN programme for *function SINORAO*.

```
      FUNCTION PRODE2(K0,IS,QE,ITH)
C  PRODE2 RETURNS THE BER OF THE CODE, FOR THE NO F/B MODE.-
      DOUBLE PRECISION DQE,FAC1,FAC2,PP,PPE,PQE,QQ,QPE,QQE,RR,SUM
      DQE=QE
      PPE=PQE=DQE/K0*(IS/2)/(K0+IS)
      QQE=1-PPE
      ITM=ITH
      IF(IS-2*ITH-1.GE.0) GO TO 100
      PQE=QQE
      ITM=IS-ITH-1
  100 QPE=1-PQE
      PRODE2=QPE/PPE
      IF(ITM.LT.0) RETURN
      PP=(1-(1-2*PPE)**K0)/2
      QQ=1-PP
      RR=1/PP-1
      FAC1=PQE*PP**IS
      FAC2=QPE*QQ**IS
      SUM=FAC1-FAC2
      IF(ITM.EQ.0) GO TO 110
      DO 120 I=1,ITM
      FAC=(IS-I+1.0)/I
      FAC1=FAC1*RR*FAC
      FAC2=FAC2/RR*FAC
  120 SUM=SUM+FAC1-FAC2
  110 PRODE2=PRODE2+SUM/PPE
      RETURN
      END
```

Figure A8.6.3: FORTRAN programme for *function PRODE2*.

A8.6.2. Probability of First Decoding Error under FD

```
      SUBROUTINE PRODE1(K0,NOC,QEQ,IDT,ERER,IAR)
C  PRODE1 RETURNS THE ERROR EXTENSION RATIO PER COSET, WITH TX F/6.-
C  CODE (N,K) = (KC+KC/NOC,KC).-
C  ON ENTRY, QEQ = PROBABILITY OF A CHANNEL ERROR / GUARANTEED ERROR
C  CORRECTING CAPABILITY.-
C  ON ENTRY, IAR(1)=I1 & IAR(2)=I2 SPECIFY THE COSETS TO BE CALCULATED:
C  I1,I1+1,...,I2.-
C  ON EXIT, QEQ = AVERAGE ERROR EXTENSION RATIO, IF I2-I1=NOC-1.-
C  IDT = DEVIATION FROM NOMINAL SYNDROME THRESHOLD.-
C  ERER(I) = ERROR EXTENSION RATIO FOR ITH COSET / I=I1,I1+1,...,I2.-
C  ON EXIT, IAR(KC+I) = NUMBER OF MESSAGE ERROR DIGITS, AFTER CORRECT
C  RESETTING, IN THE ITH SYNDROME / I=1,2,...,IS ,  ORTHOGONAL ON
C  MESSAGE DIGITS OF KCTH COSET / KC=1,2,...,NOC -
C  IAR(KC+I) < IAR(KC+I+1) ,  FOR I=1,2,...,IS.-
      DIMENSION ERER(NOC),IAR(KC),AA(200),JJ(200)
      DOUBLE PRECISION BP(200),DERE,P,P2,PRP,PRG,PPR,PQ1,GP1,RR,DEG,PGE,
     1PE,QE,QPE,SUMP,SUMQ
      COMMON/COD2/JEXP
      DEQ=QEQ
      M=K0+1
      IS=K0/NOC
      PE=PQE=DEQ/K0*(IS/2)/(KC+IS)
      ITH=(IS+1)/2
      JTH=KMX=ITH+IDT
C   CHOICE OF MODE A OR E.-
      QE=1-PE
      IF(IS-2*JTH-1.GE.0) GO TO 100
      PQE=QE
      KMX=IS-JTH-1
  100 QPE=1-PQE
      I1=IAR(1)
      I2=IAR(2)
      IF(I1.LT.1.OR.I1.GT.I2) I1=1
      IF(I2.LT.I1.OR.I2.GT.NOC) I2=NOC
      IF(KMX.GE.0) GO TO 105
      QEQ =QPE/PE
      DO 107 I=I1,I2
  107 ERER(I)=QEQ
  105 P2=1-2*PE
C   CALCULATION & REORDERING OF IAR.-
      NC=1
      KC=0
      DO 110 I=1,NOC
      MC=NC
      DO 120 J=1,IS
      MC=MOD(MC+JEXP,M)
  120 AA(J)=MC
      JJ(1)=1
      CALL ORDER(IS,AA,JJ)
      DO 130 J=1,IS
  130 IAR(J+KC)=AA(JJ(J))
      IF(I.EQ.NOC) GO TO 110
      KC=KC+IS
  140 NC=NC+1
      DO 150 J=1,KC
      IF(IAR(J).EQ.NC) GO TO 140
  150 CONTINUE
```

```
  110 CONTINUE
      KC=(-1)*IS
      QEQ=0
C   CALCULATION OF ERER.-
      DO 160 IJ=I1,I2
      KC=KC+IS
C   CALCULATION OF PRODUCTS OF P AND OF Q.-
      PRP=2.0**((-1.0)*IS)
      PRQ=PRP
      DO 170 I=1,IS
      P=1-P2**IAR(KC+I)
      BB(I)=P/(2-P)
      PRP=PRP*P
  170 PRQ=PRQ*(2-P)
      PQ1=PQE*PRP
      QP1=QPE*PRQ
      DERE=(PQ1-QP1+QPE)/PE
      IF(KMY.EQ.0) GO TO 165
      SUMP=0
      SUMQ=0
      LN=0
  180 LN=LN+1
      DO 190 I=1,LN
  190 JJ(I)=I
C   CALCULATION OF PRODUCTS OF R.-
  200 PRR=1
      DO 210 I=1,LN
  210 PRR=PRR*BB(JJ(I))
      SUMP=SUMP+1/PRR
      SUMQ=SUMQ+PRR
C   CALCULATION OF COMBINATIONS OF PRODUCTS OF R.-
      K=LN
  220 JJ(K)=JJ(K)+1
      LNK=LN-K
      IF(JJ(K).GT.IS-LNK) GO TO 230
      IF(LNK.EQ.0) GO TO 200
      DO 240 I=1,LNK
  240 JJ(K+I)=JJ(K+I-1)+1
      GO TO 200
  230 K=K-1
      IF(K.GE.1) GO TO 220
      IF(LN.LT.KMX) GO TO 180
      DERE=DERE+(PQ1*SUMP-QP1*SUMQ)/PE
  165 ERER(IJ)=DERE
  160 QEQ=QEQ+DERE
      QEQ=QEQ/NCC
      RETURN
      END
```

<u>Figure A8.6.4</u>: FORTRAN programme for *subroutine PRODE1*.

```
      SUBROUTINE ORDER(N,AA,JJ)
      DIMENSION AA(N),JJ(N)
      NNN=N/2
      MM=N+1
      IND=JJ(1)
      DO 100 I=1,N
  100 JJ(I)=I
      DO 110 I=1,NNN
      NX=I
      NN=MM-I
      DX=DDX=AA(JJ(I))
      DN=AA(JJ(MM-I))
      IF(DX.GE.DN) GO TO 120
      DX=DN
      DN=DDX
      NX=NN
      NN=I
  120 M1=JJ(NX)
      M2=JJ(NN)
      INX=INN=0
      IF(I+I.EQ.N) GO TO 170
      I1=I+1
      NI=N-I
      DO 130 J=I1,NI
      DD=AA(JJ(J))
      IF(DD.LE.DX) GO TO 140
      INX=2
      DX=DD
      NX=J
      GO TO 130
  140 IF(DD.GE.DN) GO TO 130
      INN=1
      DN=DD
      NN=J
  130 CONTINUE
  170 JJ(I)=JJ(NX)
      JJJ=JJ(NN)
      IF(I.EQ.NN) JJJ=M2
      JJ(MM-I)=JJJ
      INNX=INX+INN+1
      GO TO(110,150,160,150),INNX
  150 JJ(NN)=M2
      IF(INNX.EQ.2) GO TO 110
  160 JJ(NX)=M1
  110 CONTINUE
      IF(IND.NE.1) RETURN
      DO 180 I=1,NNN
      JH=JJ(I)
      JJ(I)=JJ(MM-I)
  180 JJ(MM-I)=JH
      RETURN
      END
```

Figure A8.6.5: FORTRAN programme for *subroutine ORDER*.

APPENDIX 8.7: SELECTION OF CODES WITH GIVEN PARAMETERS

In this appendix, three algorithms will be presented. For each one of them, given two code parameters the routine returns the (k,J) type-C5 code with one parameter matched exactly and the other being as close as possible.

A8.7.1. Codes with Given Information-Block Length, k

This function [IORD1(k,J) $\hat{=}$ $f1$]* returns $f1$ so that (k,$f1$) is a type-C5, or type-B4 code, with $f1$ as close to J as possible. The algorithm (Fig. A8.7.1) is original.

If $k+1 \le 2$, there is no code and $f1=0$. If k+1=even there is only a type-B4 code, with $f1=2$. For the rest of the cases, $f1$ must divide $\theta(k+1)$. Since k is fixed, $\theta(k+1)$ will determine the solution. If $J \le 2$, then $f1=2$ [$\theta(k+1)$ is even]. If $J \ge \theta(k+1)$ then $f1=\theta(k+1)$. If $\theta(k+1) \le 3$, then $f1 = \theta(k+1)$. For the rest of the cases, test $f1$s are J, J-1, J+1, J-2, J+2, etc. A solution will be found eventually, because the sequence of test $f1$s starts with a $f1 \in (2,\theta(k+1))$, hence it will terminate either with $f1=2$, or with $f1=\theta(k+1)$, both of which are valid solutions.

```
If (k < 2), then:    ƒ1 = 0  ---------------------------------> END
If (k+1 = even  or  J ≤ 2), then:   ƒ1 = 2  ------------------> END
If [θ(k+1) ≤ 3  or  θ(k+1) ≤ J], then:   ƒ1 = θ(k+1)  -------> END
    ƒ1 = J  &  δ = 0  &  s = -1
┌-> Step:  If [ƒ1 | θ(k+1)], then:  ---------------------------> END
│          If else, then:  s = -s  &  δ = δ+1  &  ƒ1 = ƒ1+s×δ
└--------  Repeat step
```

Figure A8.7.1: Flow-chart for given k and nearest J.

A8.7.2. Codes with Given Rate, c/(c+1)

This function [IORD4(c,k) $\hat{=}$ $f4$]** returns $f4$ so that ($cf4$,$f4$) is a type-C5, or type-B4 code, with $c \times f4$ as close to k as possible. The algorithm (Fig. A8.7.2) is original.

Because $k = c \times J$ = even, if c = odd, then J must be even. Also, since $J \ge 2$, then $k/c \ge 2$. The first candidate for $f4$ is $\lfloor k/c \rfloor$; if this is odd and c is also odd, it is reduced by 1. The next candidate will be $f4+s \times \delta$, where $s = \pm 1$. Before a new value of $f4$ is generated, s changes sign and δ is in-

* See Appendix 8.8 (§ A8.8.1., p. 566).
** See Appendix 8.8 (§ A8.8.2., p. 567).

creased by 1 if c=even, or 2 if c=odd. Since the information block length is c×ƒ4, the test for each candidate is ƒ4 | θ(c×ƒ4+1). The search will terminate at least with ƒ4=2.

```
     ƒ4 = MAX{2,INT(k/c)}  &  Md = c mod 2
     If (c & ƒ4 are odd), then:  ƒ4 = ƒ4-1
     s = Sign(c×ƒ4+0.5-k)  &  δ = 0
┌─>  Step:  If [ƒ4 | θ(c×ƒ4+1)], then:  ─────────────> END
│           s = -s  &  δ = δ+1+Md  &  ƒ4 = ƒ4+s×δ
└────────   If (ƒ4 > 2), then:  Repeat step
            If else, then:  ƒ4 = 2  ─────────────────> END
```

Figure A8.7.2 Flow-chart for given code-rate and nearest k.

A8.7.3. Codes with Given Number of Orthogonal Checks, J

This function [IORD6(J,k) ≙ ƒ6]* returns ƒ6 so that (ƒ6,J) is a type-C5, or type-B4 code, with ƒ6 as close to k as possible. The algorithm (Fig. A8.7.3) is original.

If J < 2, ƒ6 = 0. If J = 2, there is either a k type-B4 code (if k=odd), or a (k,2) type-C5 (if k=even). For the rest of the cases, J≥3. To avoid a very long search, an upper limit, KMAX, is placed upon ƒ6. Also, ƒ6 must be even and a multiple of J. If the latter is odd, then q = ƒ6/J = even. If δ = J for J=even & δ = 2J for J=odd, then ƒ6 = q×δ ≤ KMAX /q=1,2,... Hence, the maximum value of q is Kmx ≙ ⌊KMAX/δ⌋. So, the candidates for ƒ6 are q×δ /q=1,2,...,Kmx.

```
If (J < 2), then:  ƒ6 = 0  ──────────────────────────────> END
If (J = 2), then:  ƒ6 = MIN{MAX[2,k],KMAX}  ─────────────> END
If (J ≥ 3), then:  δ = [1+MOD(J,2)]J  &  Kmx = INT(KMAX/δ)
                   q = INT(k/δ+0.5)  &  q = MIN{MAX[1,q],Kmx}
                   s = Sign(q×δ-k-0.5)  &  i = 0
┌─> Step 1:  If [J | θ(q×δ+1)], then:  ƒ6 = q×δ  ──────────> END
│            If else, then:  i = i+1  &  s = -s  &  q = q+s×i
└──────────  If (1 ≤ q ≤ Kmx), then:  Repeat step 1
             If (q > Kmx), then:  Go to step 3  ──────┐
             If (q < 1), then:  q = q+i               │
┌─> Step 2:  q = q+1  &  If [J | θ(q×δ+1)], then: ƒ6 = q×δ ─┼──> END
│            If (else & q > Kmx), then:  ƒ6 = 1  ──────┼─> END
└──────────  If else, then:  Repeat step 2            │
    Step 3:  q = q-i  <───────────────────────────────┘
┌─> Step 4:  q = q-1  &  If [J | θ(q×δ+1)], then:  ƒ6 = q×δ ──> END
│            If (else & q < 1), then:  ƒ6 = 1  ──────────> END
└──────────  If else, then:  Repeat step 4
```

Figure A8.7.3: Flow-chart for given J and nearest k.

* See Appendix 8.8 (§ A8.8.3., p. 568).

The test is J | θ(ƒ6+1). The first candidate is the multiple of δ, closest to k: ƒ6 = qδ, with q = ⌊k/δ+0.5⌋. Thereafter, q is decreased by 1 if qδ > k, or increased by 1 if otherwise and the search considers q±1, q±2, etc. If q becomes less than 1, only higher values are considered. If q becomes greater than Kmx, only smaller values are considered. If no suitable value is found, ƒ6 = 1.

APPENDIX 8.8: FORTRAN PROGRAMMES FOR APPENDIX 8.7

A8.8.1. Codes with Given Information-Block Length, k

```
      FUNCTION IORD1(IMO,IS)
C  IORD1 = CLOSEST TO IS INTEGER, SUCH THAT, IORD1>1, AND IORD1 DIVIDES
C IL=G.C.D.( IPR(1)-1,IPR(2)-1,...,IPR(NR)-1 ), WHERE IPR(I)/I=1,2,...,
C NR, ARE THE PRIME DIVISORS OF IMO.-
C  IF THERE IS NO SOLUTION, IORD1=1 - IF IMO<3, IORD1=0.-
      DIMENSION KAR(10,2)
      COMMON/ORD/KAR,IL,NR
      IF(IMO.GE.3) GO TO 200
      IORD1=0
      RETURN
  200 IF(MOD(IMO,2).NE.0) GO TO 230
  210 IORD1=2
      RETURN
  230 CALL PRIDE2(IMO,IL,NR,KAR)
      IORD1=IL
      IF(IL.LE.3) RETURN
      IF(IS.GE.IL) RETURN
      IF(IS.LE.2) GO TO 210
      IORD1=IS
      INC=0
      ISI=-1
  310 IF(MOD(IL,IORD1).EQ.0) RETURN
      ISI=(-1)*ISI
      INC=INC+1
      IORD1=IORD1+ISI*INC
      GO TO 310
```

Figure A8.8.1: FORTRAN programme for *function IORD1*.

A8.8.2. Codes with Given Rate, c/(c+1)

```
      FUNCTION IORD4(IR1,IR2,N)
C  A) IF: IR1>IR2-1, OR 2*IR1<IR2, THEN IORD4=0.-
C  B) IF: IR2-IR1>2*(IR1,IR2), THEN IORD4=1.-
C  C) IF: IR2-IR1=2*(IR1,IR2), THEN IORD4=2.-
C  D) IF: IR2-IR1=(IR1,IR2), THEN IORD4 = CLOSESEST TO N*(IR2-IR1)/IR1
C INTEGER, SUCH THAT: 1) IORD4>1.-
C       2) IORD4 DIVIDES IL, WHERE IL=G.C.D.( IPR(1)-1,IPR(2)-1,...,
C          IPR(NR)-1 ), AND IPR(I)/I=1,2,...,NR,  ARE ALL THE PRIME DI-
C          VISORS OF IR1/(IR1,IR2)*IORD4+1.-
C   AND 3) IF 2 INTEGERS, SATISFYING CONDITIONS (1) AND (2), ARE EQUI-
C          DISTANT FROM N, THE LOWER IS CHOSEN AS IORD4.-
C  IN OTHER WORDS: IORD4 RETURNS THE NO OF ORTHOGONAL CHECK SUMS J, FOR
C A CODE OF RATE R=IR1/IR2 AND BLOCK CONSTRAINT LENGTH AS CLOSE TO N AS
C POSSIBLE, WITH PREFERENCE TO LOWER VALUES - IN PARTICULAR, J=0 IF THE
C VALUE OF R IS ILLEGAL, AND J=1 IF THERE IS NO CODE FOR THIS R.-
      DIMENSION KAR(22,2)
      IF(IR1.LT.IR2.AND.2*IR1.GE.IR2) GO TO 290
      IORD4=0
      RETURN
  290 IC=IGCD(IR1,IR2)
      JR1=IR1/IC
      JR2=IR2/IC
      IF(JR2.EQ.JR1+1) GO TO 350
      IORD4=1+2/(JR2-JR1)
      RETURN
  350 IORD4=MAX0(N,2*JR1)/JR1
      IV=MOD(JR1,2)
      IW=IV+MOD(IORD4,2)
      IORD4=IORD4-IW/2

      ISI=ISIGN(1,2*(IORD4*JR1-N)+1)
      IM=0
  410 CALL PRIDE2(JR1*IORD4+1,IL,NR,KAR)
      IF(MOD(IL,IORD4).EQ.0) RETURN
      ISI=(-1)*ISI
      IM=IM+1+IV
      IORD4=IORD4+ISI*IM
      IF(IORD4.GT.2) GO TO 410
      IORD4=2
      RETURN
      END
```

Figure A8.8.2: FORTRAN programme for *function IORD4.*

A8.8.3. Codes with Given Number of Orthogonal Checks, J

```
      FUNCTION IORD6(J,N)
C  A) IF: J<2, IORD6=0.-
C  B) IF: J=2 AND N<2, IORD6=2.-
C  C) IF: J=2 AND N<MAXN+1, IORD6=MAXN.-
C  D) IF: J=2 AND 1<N<MAXN+1, IORD6=N.-
C  E) IF: J>2 THEN IORD6=CLOSEST TO N INTEGER SUCH THAT:
C   1) J DIVIDES IL, WHERE IL=G.C.D.( IPR(1)-1,IPR(2)-1,...,IPR(NR)-1 ),
C    AND IPR(I)/I=1,2,...,NR, ARE ALL THE PRIME DIVISORS OF IORD6+1.-
C   2) IF TWO INTEGERS, SATISFYING CONDITION (1), ARE EQUIDISTANT FROM
C    J, THE LARGER IS CHOSEN AS IORD4.-
C   3) IF THERE IS NO INTEGER, SATISFYING THE ABOVE CONDITIONS, IN THE
C    RANGE [1,MAXN], IORD6=1, WHERE MAXN IS EVALUATED IN 310.-
C  IN OTHER WORDS, IORD6 RETURNS THE BLOCK-CONSTRAINT LENGTH, FOR A CO-
C DE WITH J ORTHOGONAL CHECK-SUMS AND BLOCK-CONSTRAINT LENGTH, AS CLOSE
C AS POSSIBLE TO N, WITH PREFERENCE TO HIGHER VALUES - IN PARTICULAR,
C IORD6=0,IF THE VALUE OF J IS ILLEGAL, AND IORD6=1 IF THERE IS NO CODE
C WITH NUMBER OF ORTHOGONAL CHECK SUMS J AND BLOCK CONSTRAINT LENGTH
C LESS THAN MAXN+1.-
      INTEGER X02BBF
      DIMENSION KAR(22,2)
      COMMON/TR6/MAXN
  310 MAXN=9999
      IORD6=0
      IF(J.LT.2) RETURN
      KX=J*(MOD(J,2)+1)
      NN=MIN0(MAX0(J,N),(MAXN/KX+1-1/(1+MOD(MAXN,KX)))*KX/2-1)
      IORD6=NN
      IF(J.EQ.2) RETURN
      IORD6=KX*INT(FLOAT(NN)/KX+0.5)
      JR=ISIGN(1,2*(IORD6-NN)-1)
      JS=0
      JA=1
      JB=-1
      JN=0
      JNM=2*IORD6/KX-(JR+3)/2
  450 IMO=IORD6+1
      CALL PRIDE3(IMO,IL,NR,KAR)
      IF(MOD(IL,J).EQ.0) RETURN
      IF(JN.GE.JNM) GO TO 540
  490 JN=JN+1
      JS=JA*JS+1
      JR=JB*JR
      IORD6=IORD6+JR*JS*KX
      GO TO 450
  540 IF(JA.NE.1) GO TO 590
      JA=0
      JB=1
      JNM=MAXN/KX-1
      GO TO 490
  590 IORD6=1
      RETURN
      END
```

Figure A8.8.3: FORTRAN programme for *function IORD6*.

APPENDIX 8.9: CONFIDENCE INTERVALS

Consider a sample of size n, and let m denote the *sample mean*. Then, if $n \geq 30$, the sample means, m, are normally-distributed random variables with mean, say, μ_m and standard deviation, say, σ_m (see Erricker [49], p. 196). It can also be shown (ibid, pp. 196-9) that,

$$\mu_m = \mu \qquad (A8.9.1)$$

and that

$$\sigma_m = \sigma/\sqrt{n} \qquad (A8.9.2)$$

where μ & σ are the population parameters.

Of course, μ & σ are not known, but they can be estimated. As mentioned earlier, the best estimate of μ is the sample mean, m. The best estimate of σ^2 is $ns^2/(n-1)$ (see Erricker [49], p. 226), where s^2 is the variance of the sample,

$$s^2 = (1/n)\sum_{i=1}^{n}(x_i-m)^2 \qquad (A8.9.3)$$

and x_i are the sample values.

From the graphs of the normal probability density function, it may be deduced that:

99% of the area lies between $\mu - 2.58\sigma$ & $\mu + 2.58\sigma$
95% of the area lies between $\mu - 1.96\sigma$ & $\mu + 1.96\sigma$
90% of the area lies between $\mu - 1.645\sigma$ & $\mu + 1.645\sigma$
50% of the area lies between $\mu - 0.6745\sigma$ & $\mu + 0.6745\sigma$

Hence, if σ_m is known, one may state that μ lies between $m-2.58\sigma_m$ & $m+2.58\sigma_m$, with confidence 99% (i.e. 99% of the sample means m, lie in $m\pm2.58\sigma_m$).

From (A8.9.3):

$$s^2 = (1/n)\sum_{i=1}^{n}x_i^2 - 2m(1/n)\sum_{i=1}^{n}x_i + m^2(1/n)\sum_{i=1}^{n}1 \longrightarrow$$

$$s^2 = (1/n)\sum_{i=1}^{n} x_i^2 - 2m^2 + m^2 \quad \Longrightarrow$$

$$s^2 = (1/n)\sum_{i=1}^{n} x_i^2 - m^2 \quad \Longrightarrow \quad [\text{since } x_i = 0 \text{ or } 1 \text{ - bit stream}]$$

$$s^2 = (1/n)\sum_{i=1}^{n} x_i - m^2 = m - m^2 \quad \Longrightarrow$$

$$s^2 = m(1-m) \qquad (A8.9.4)$$

Then: $$\sigma^2 = nm(1-m)/(n-1) \qquad (A8.9.5)$$

From (A8.9.2) & (A8.9.5), since m, the sample mean, is the estimate of the probability P_x of the bit stream (x = e or d), and since the sample size is very large (of the order of 10^4 , or more), $n-1 \approx n = Nb$:

$$\sigma_m = \sqrt{[\tilde{P}_x(1-\tilde{P}_x)/Nb]} \qquad (A8.9.6)$$

(A8.9.6) may be used to obtain the 99% confidence intervals in estimating P_x.

APPENDIX 8.10: GRAPHS OF EER & NET CODING-GAIN vs Γ*

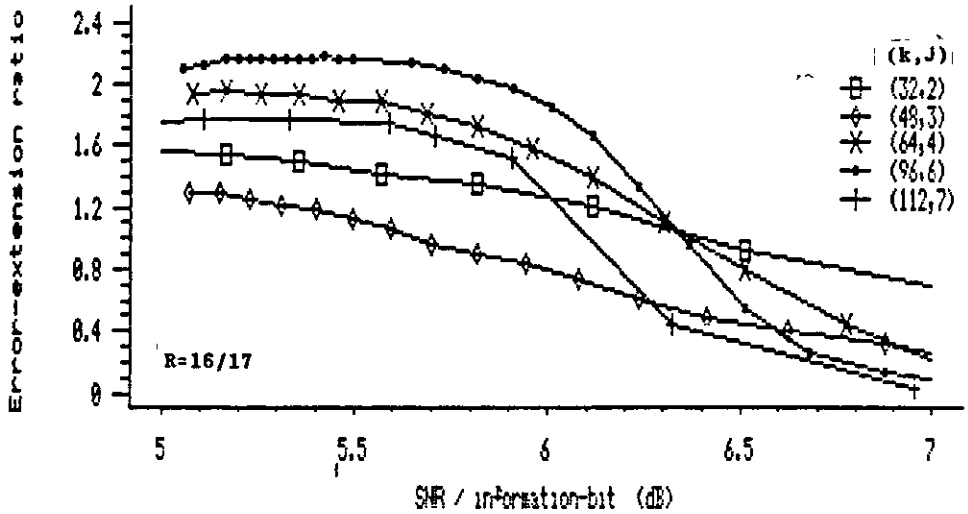


Figure A8.10.1: EER vs Γ, for rate 16/17 codes. *

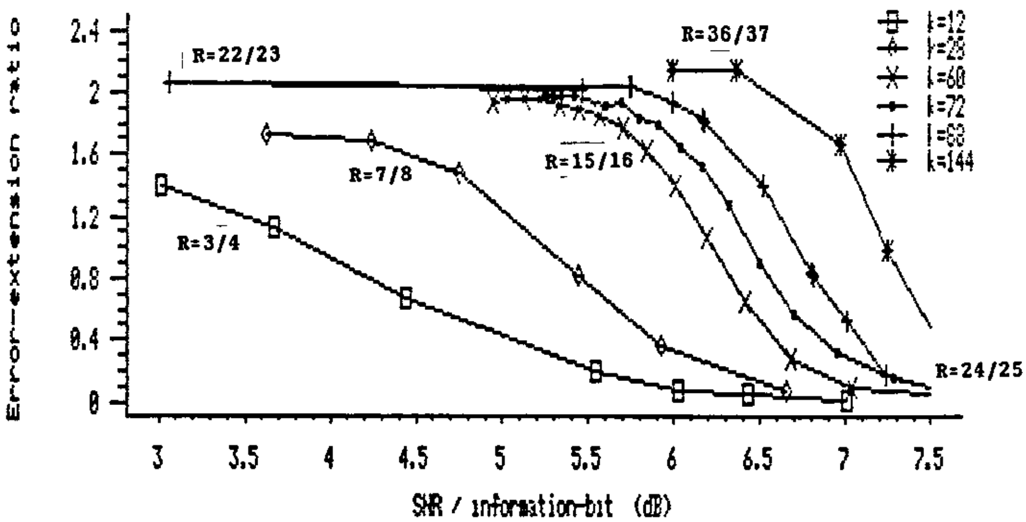
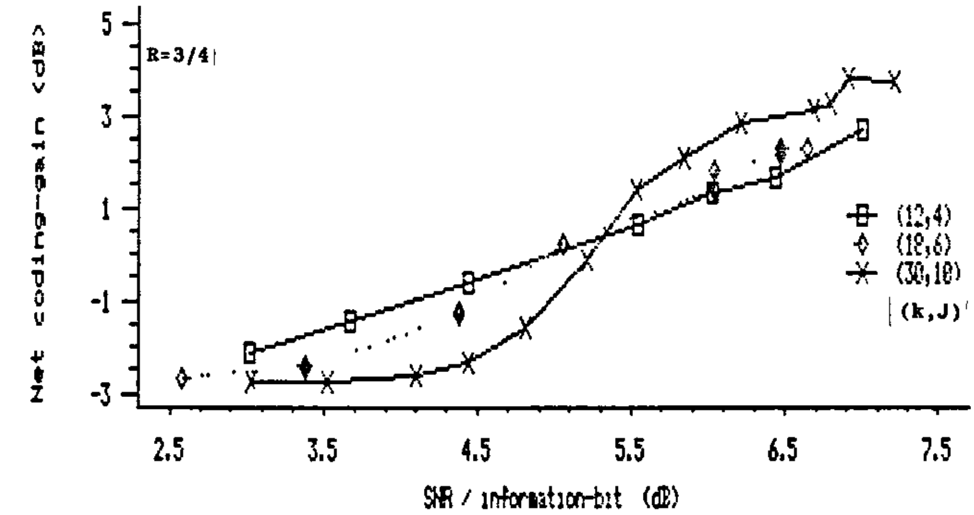
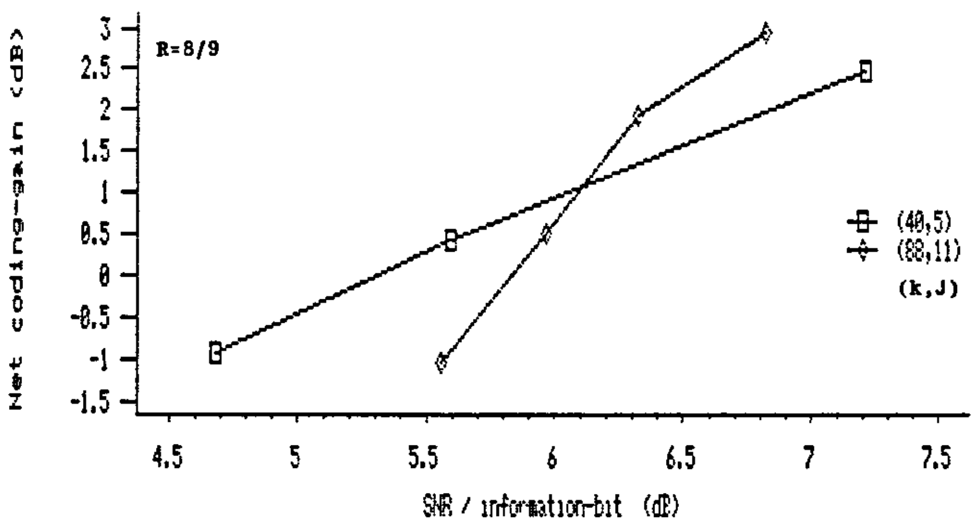


Figure A8.10.2: EER vs Γ, for J = 4 codes. *

* EER = error-extension ratio.
Γ = signal-to-noise power-ratio per information-bit.





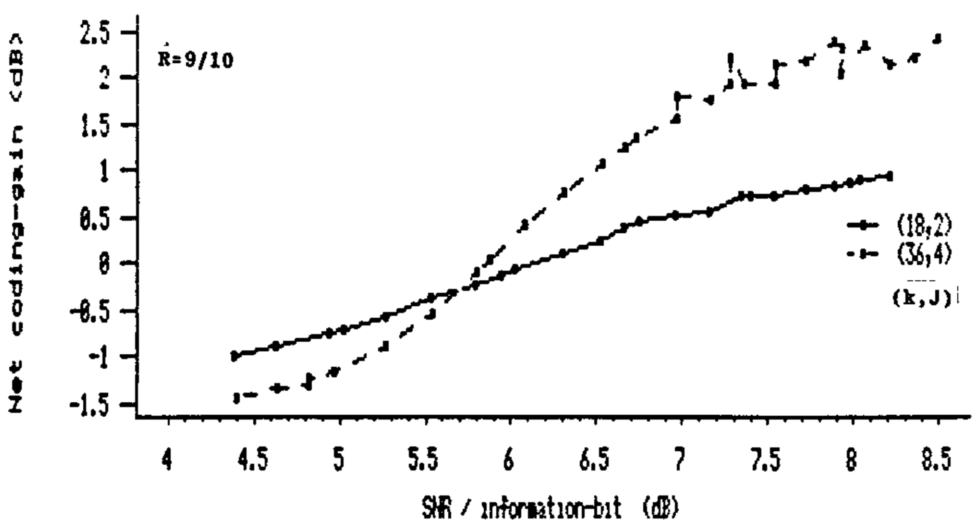


Figure A8.10.3: Net coding-gain vs Γ for codes of rate 3/4 (top), R=8/9 (middle) & R=9/10 (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

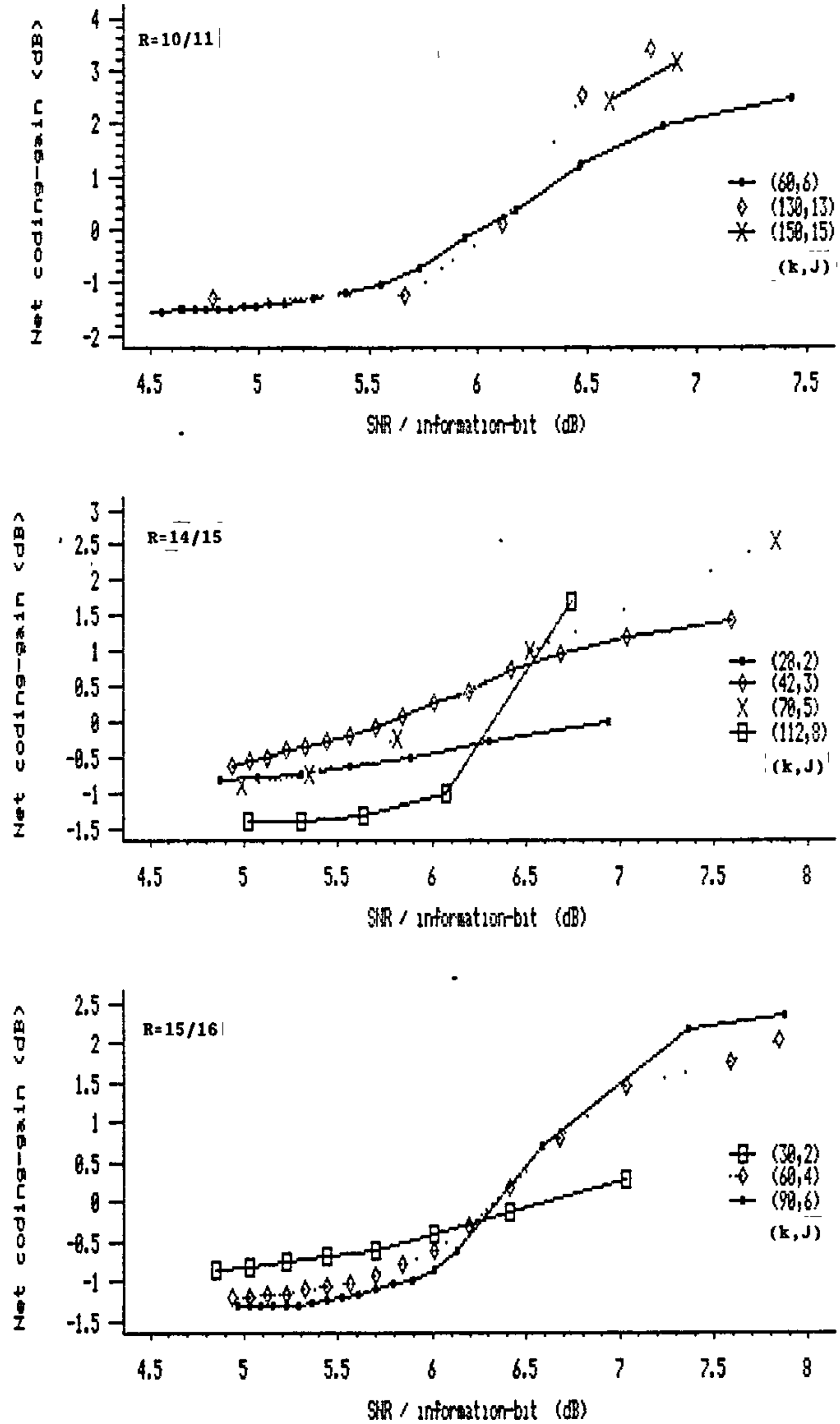
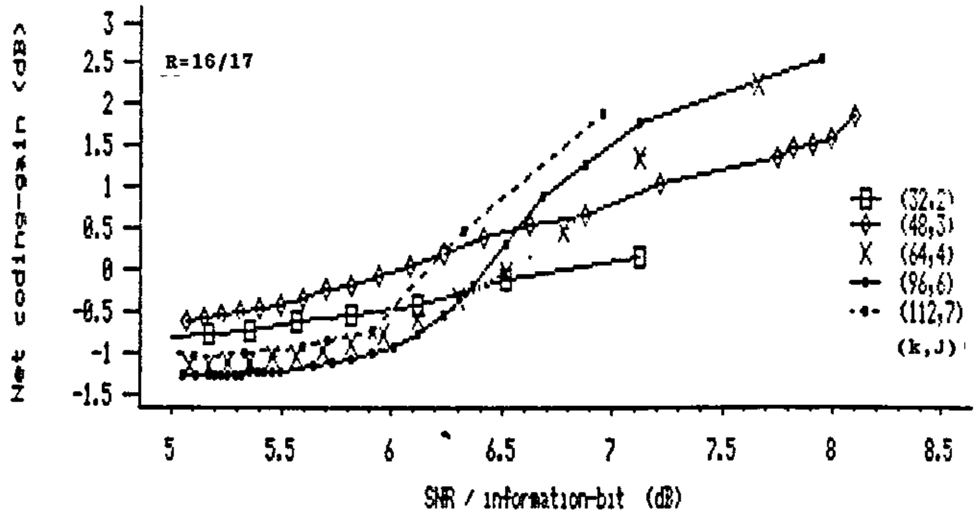


Figure A8.10.4: Net coding-gain vs Γ for codes of rate 10/11 (top), R=14/15 (middle) & R=15/16 (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.



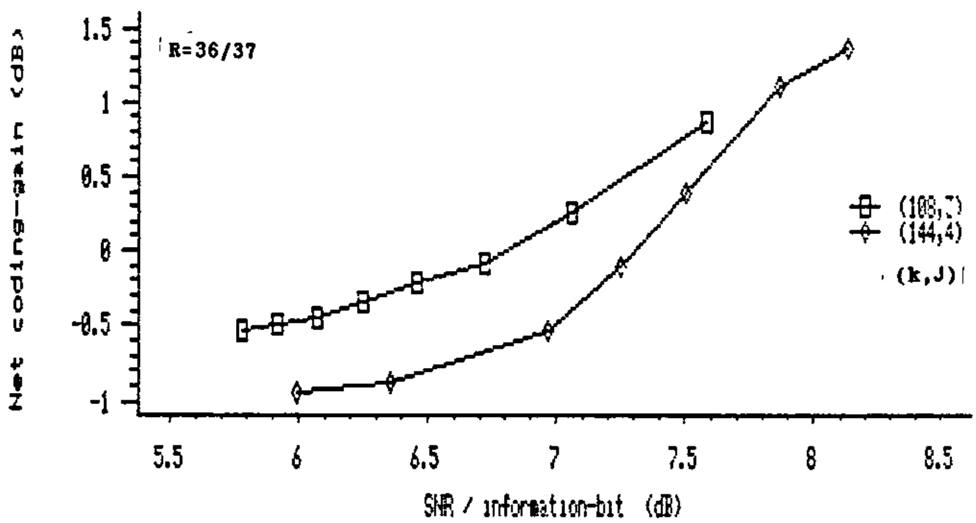


Figure A8.10.5: G vs Γ for R=16/17 (top) & R=36/37, codes. *

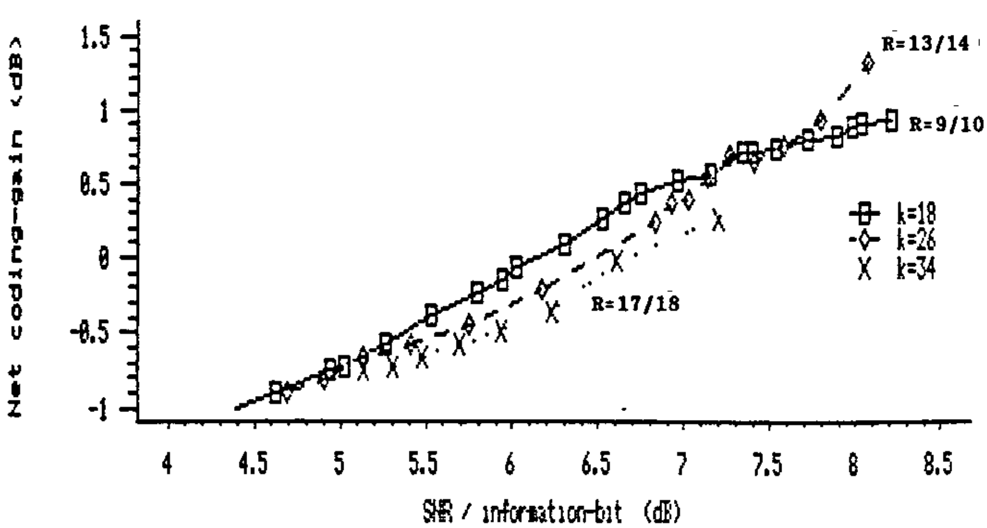


Figure A8.10.6: Net coding-gain vs Γ for J = 2 codes.*

* Γ = signal-to-noise power-ratio per information-bit.

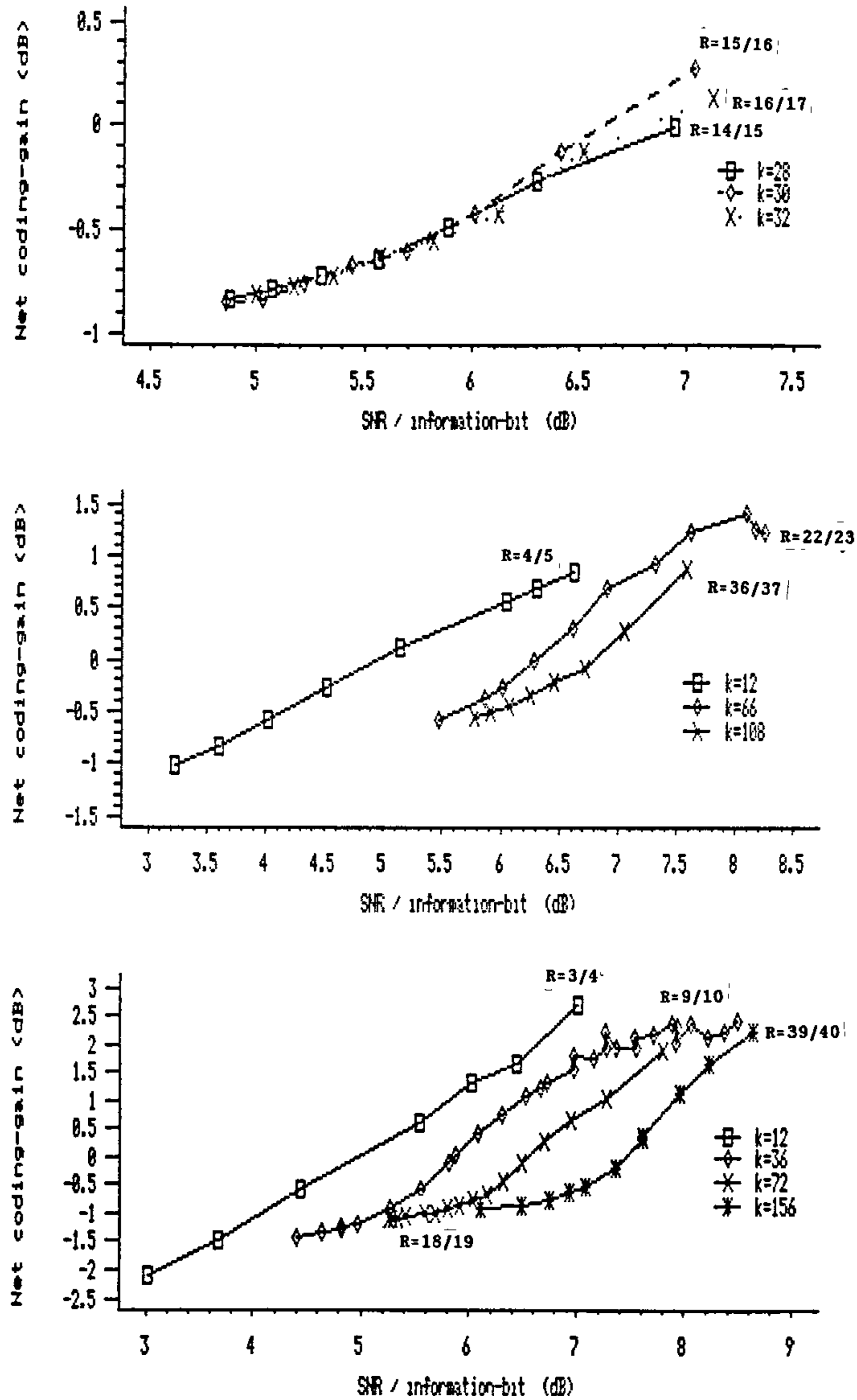
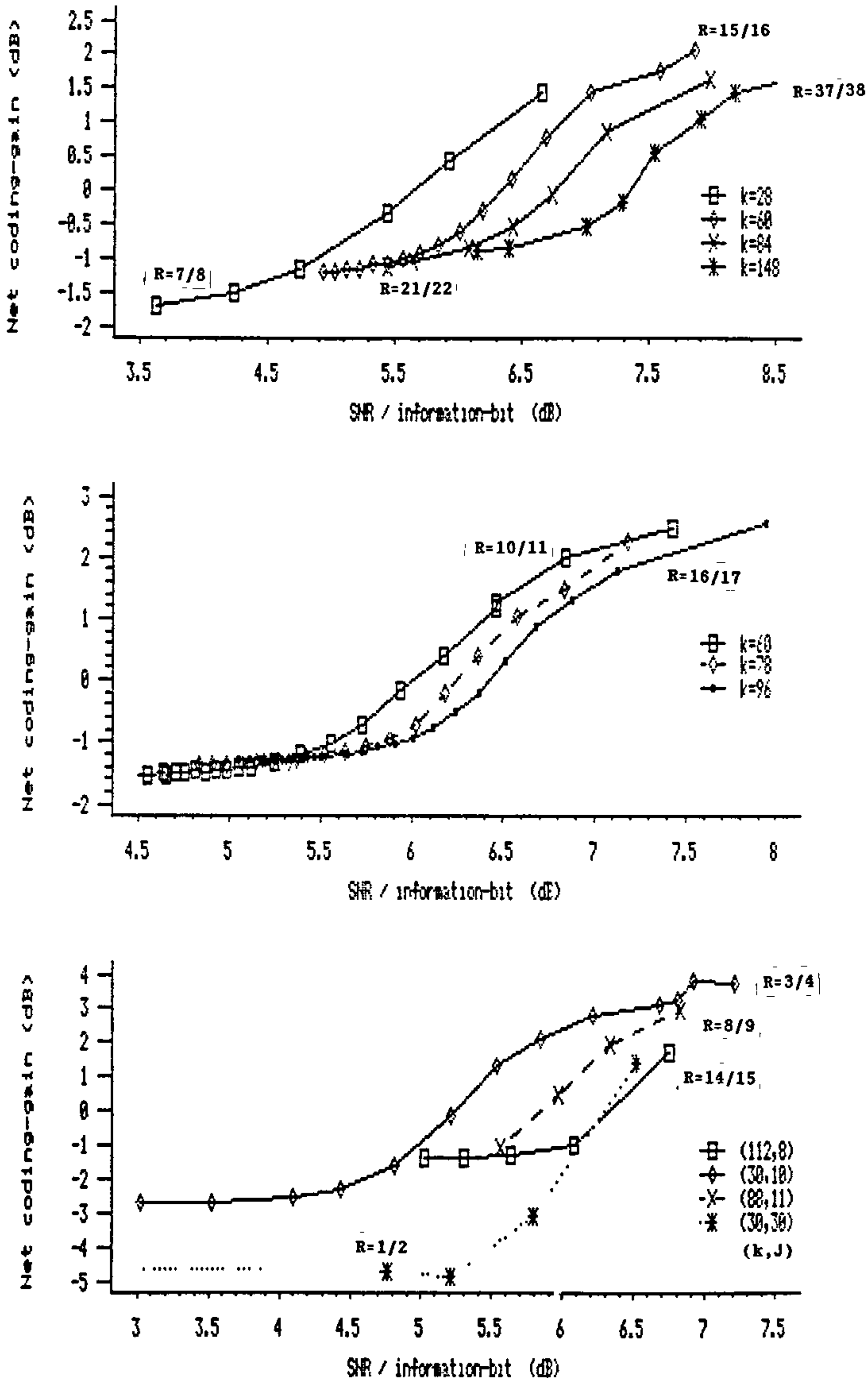


Figure A8.10.7: Net coding-gain vs Γ for codes with J=2 (top), J=3 (middle) & J=4 (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.



<u>Figure A8.10.8</u>: Net coding-gain vs Γ for codes with J=4 (top), J=6 (middle) & higher Js (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

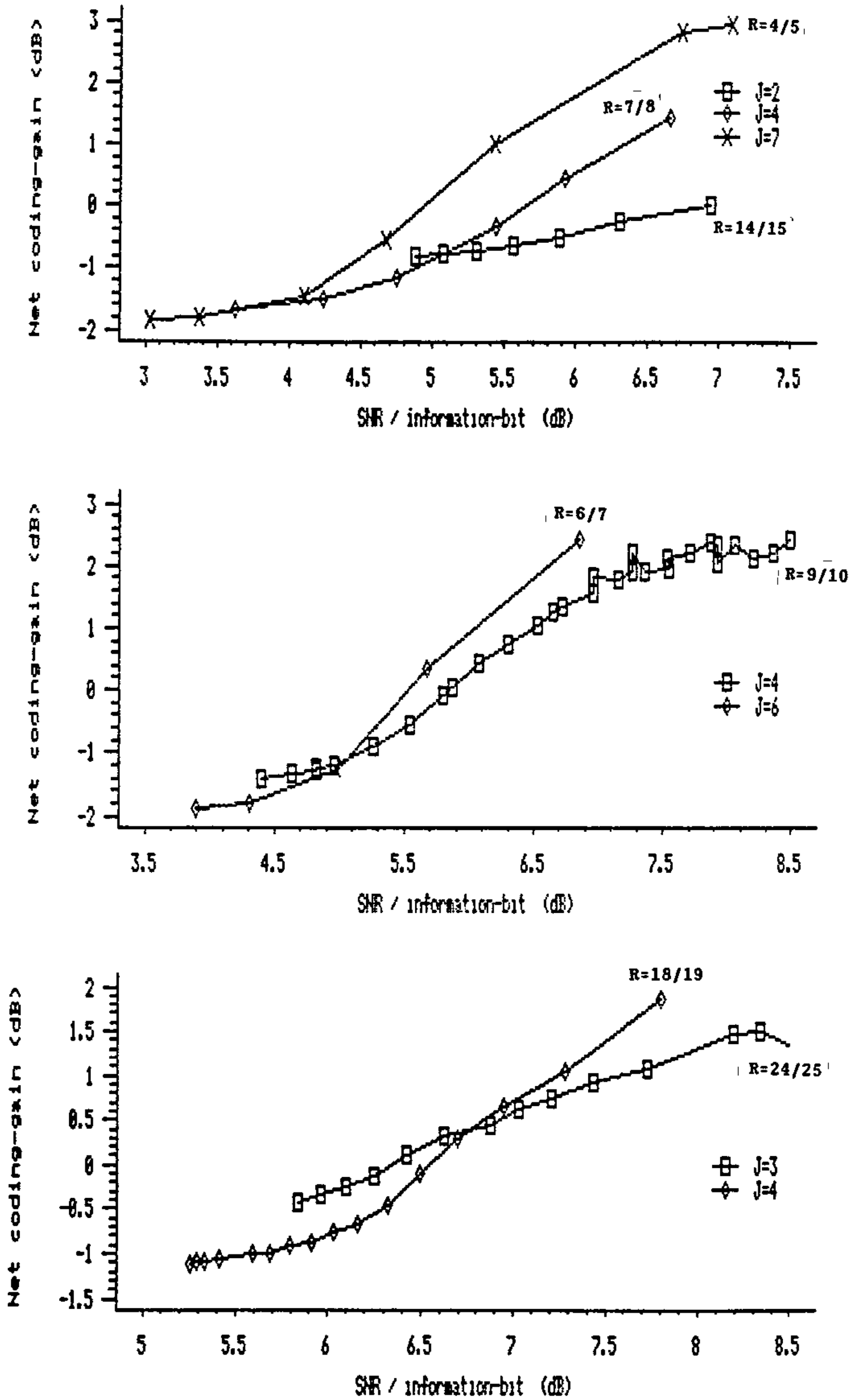


Figure A8.10.9: Net coding-gain vs Γ for codes with k=28 (top), k=36 (middle) & k=72 (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

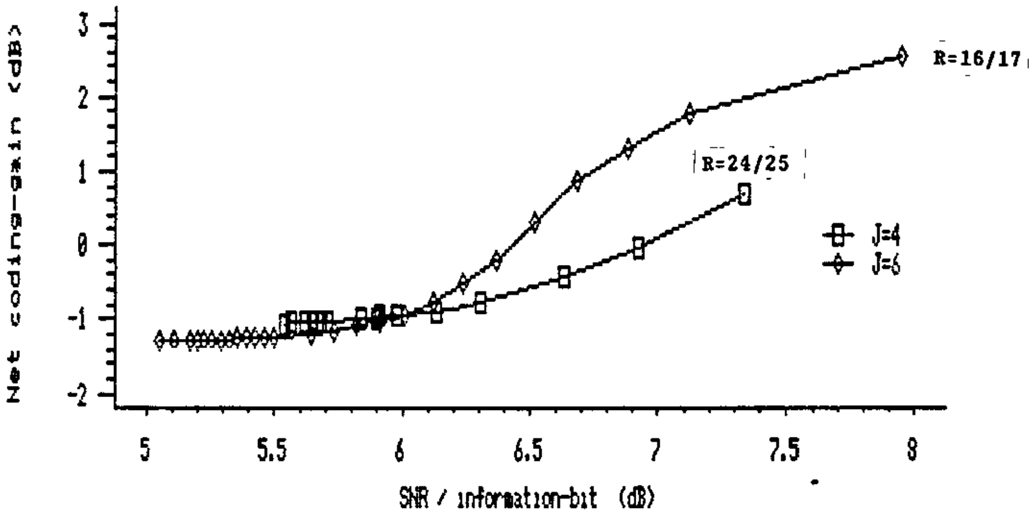


Figure A8.10.10: Net coding-gain vs Γ for k=96 codes. *

APPENDIX 8.11: ERROR PROPAGATION

TX f/b (*transmitter feedback*) denotes what has been called 'genie' decoding.** Under this mode, the syndrome register is reset using the true values of the error bits, instead of the estimated ones. *DE f/b* (*decoder feedback*) denotes normal FD. The *decoder-output error-sequence* is the number of decoding errors per b (k-bit) blocks, where b ≥ 1.

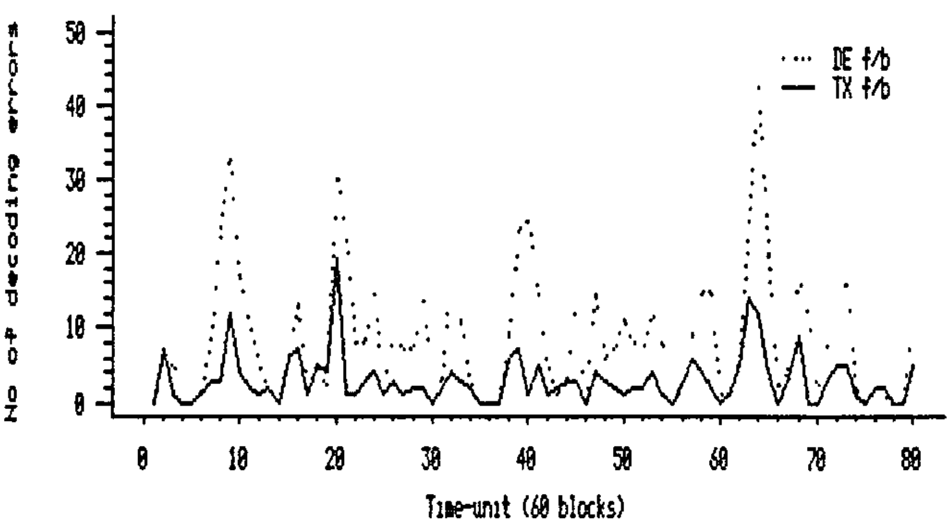


Figure A8.11.1: The decoder-output error-sequence (60 blocks per element), of the (60,6) code, at QE=5, with DE & TX f/b.

* Γ = signal-to-noise power-ratio per information-bit.
** See § 6.1.4. (p. 157).

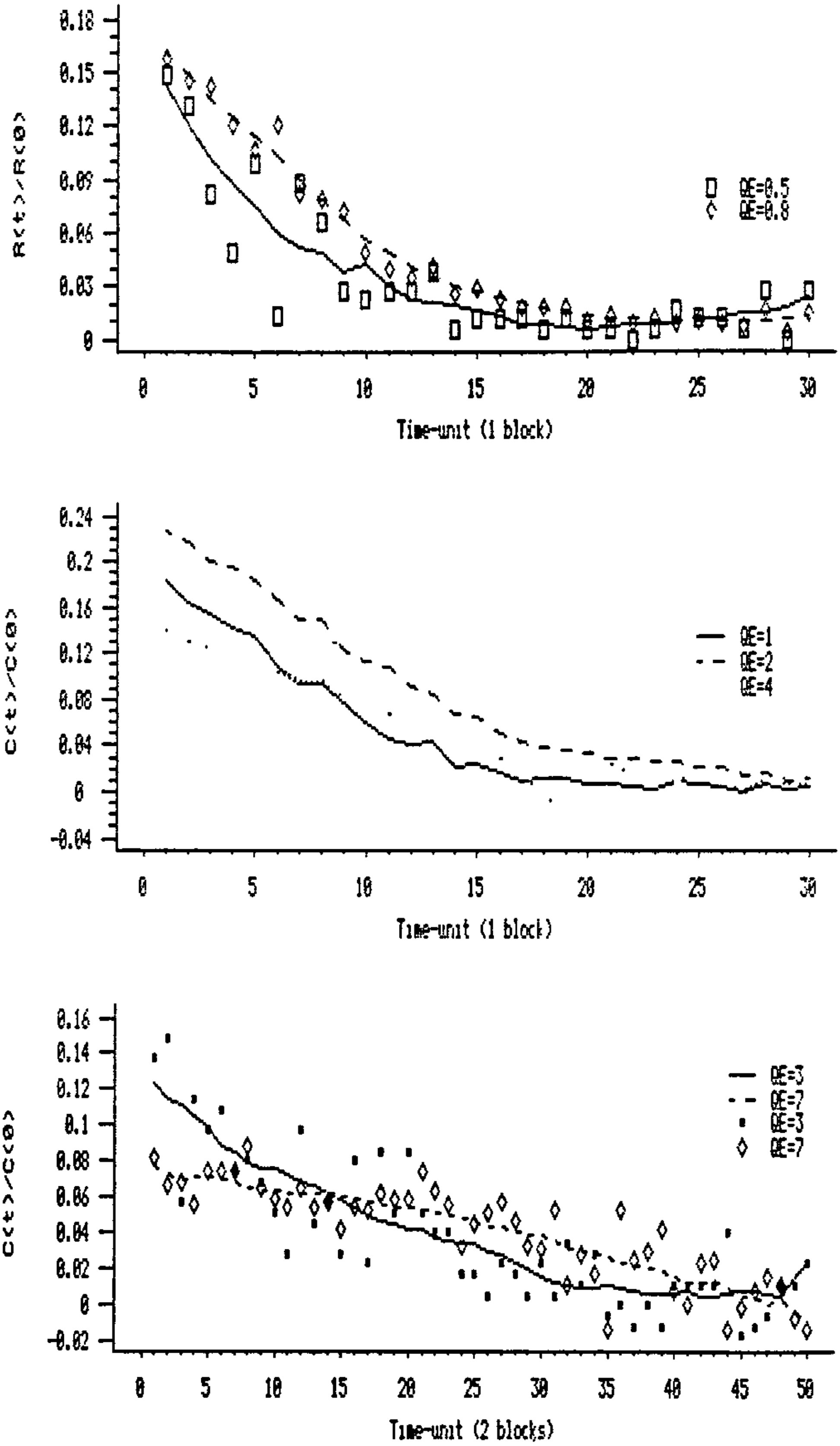


Figure A8.11.2: Autocorrelation fn (top) & autocovariance fn (middle) for the (16,4) code, with b=1. Autocov. fn for the (84,4) code, with b=2 (bottom). *

* All fns are normalized.

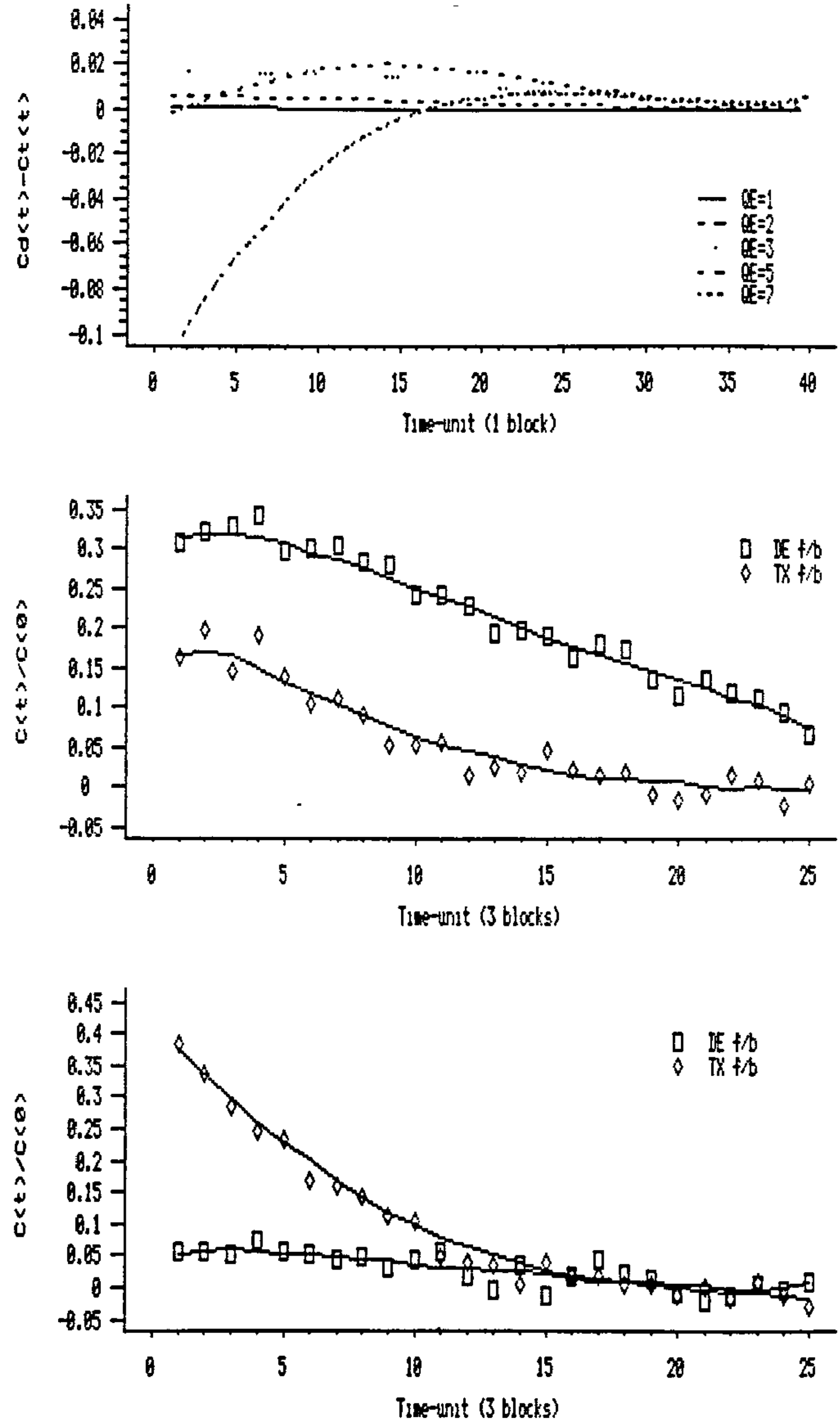


Figure A8.11.3: $C_D(\tau)-C_T(\tau)$ for the (28,4) code, with b=1 (top). $C_n(\tau)$ for the (60,6) code with TX & DE f/b and b=3; QE=5 (middle), QE=10 (bottom).

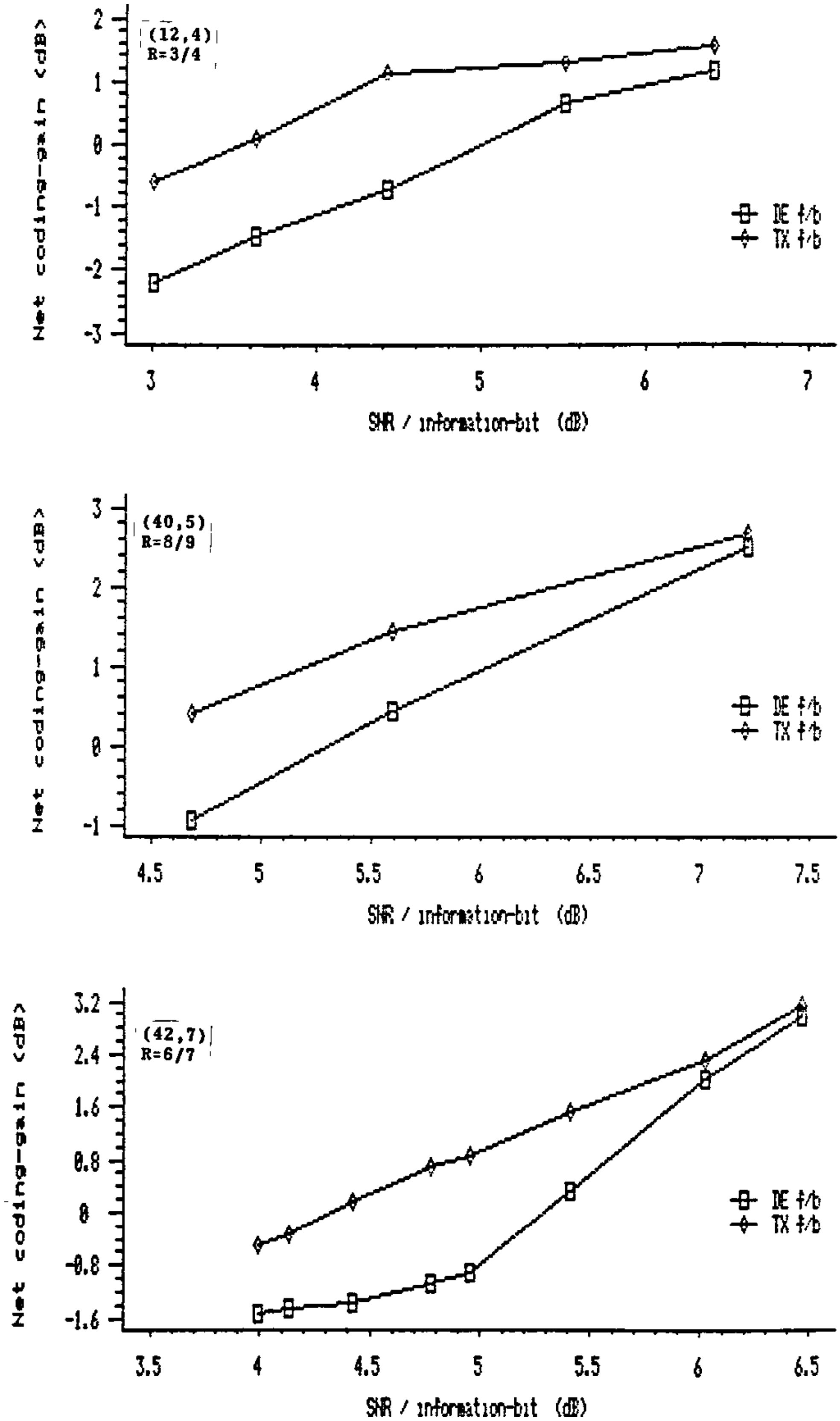


Figure A8.11.4: Net coding-gain vs Γ, with DE & TX f/b, for the (12,4) code (top), the (40,5) code (middle) & the (42,7) code (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

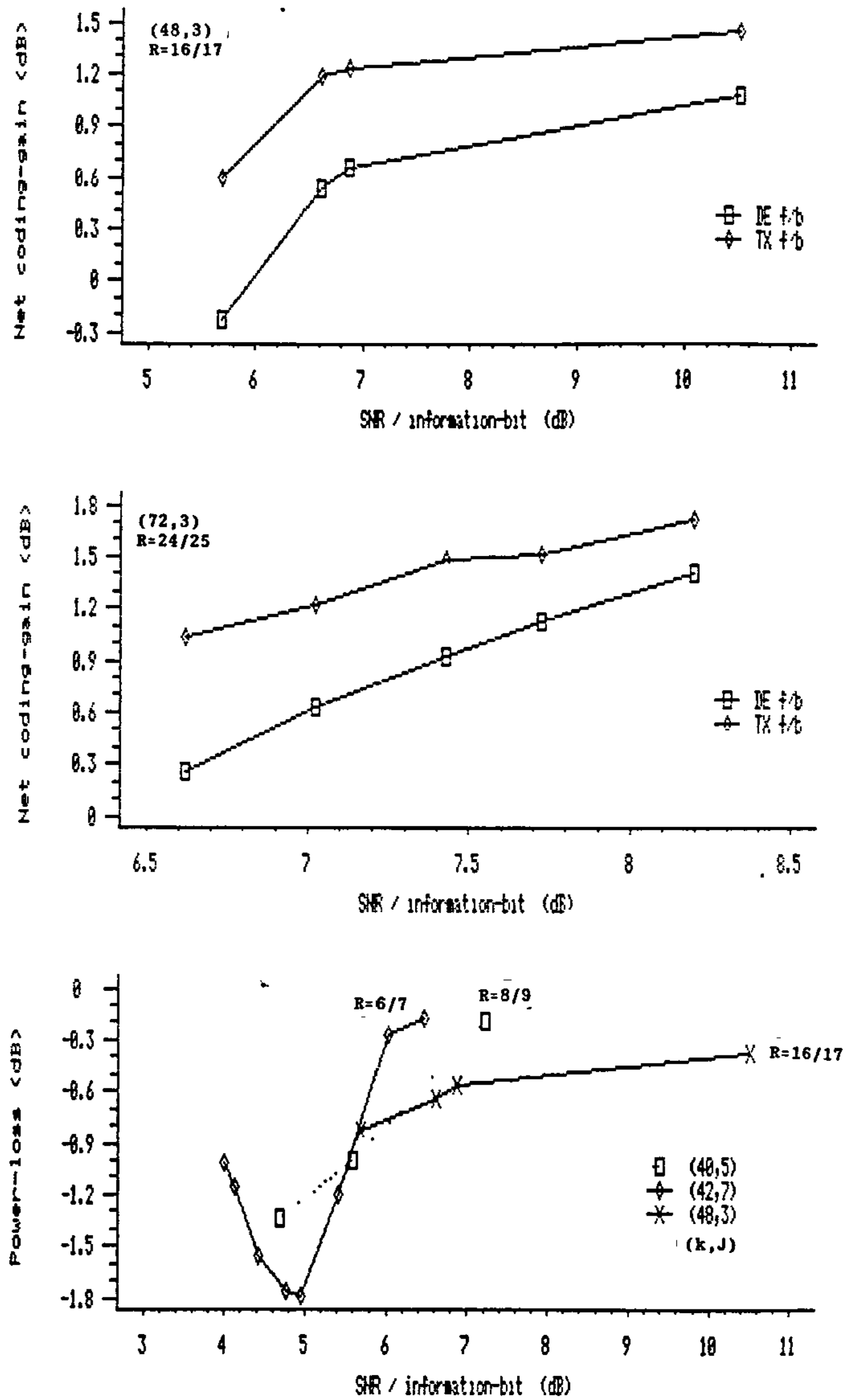


Figure A8.11.5: Net coding-gain vs Γ, with DE & TX f/b, for the (48,3) code (top) & the (72,3) code (middle). Power-loss due to error-propagation (bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

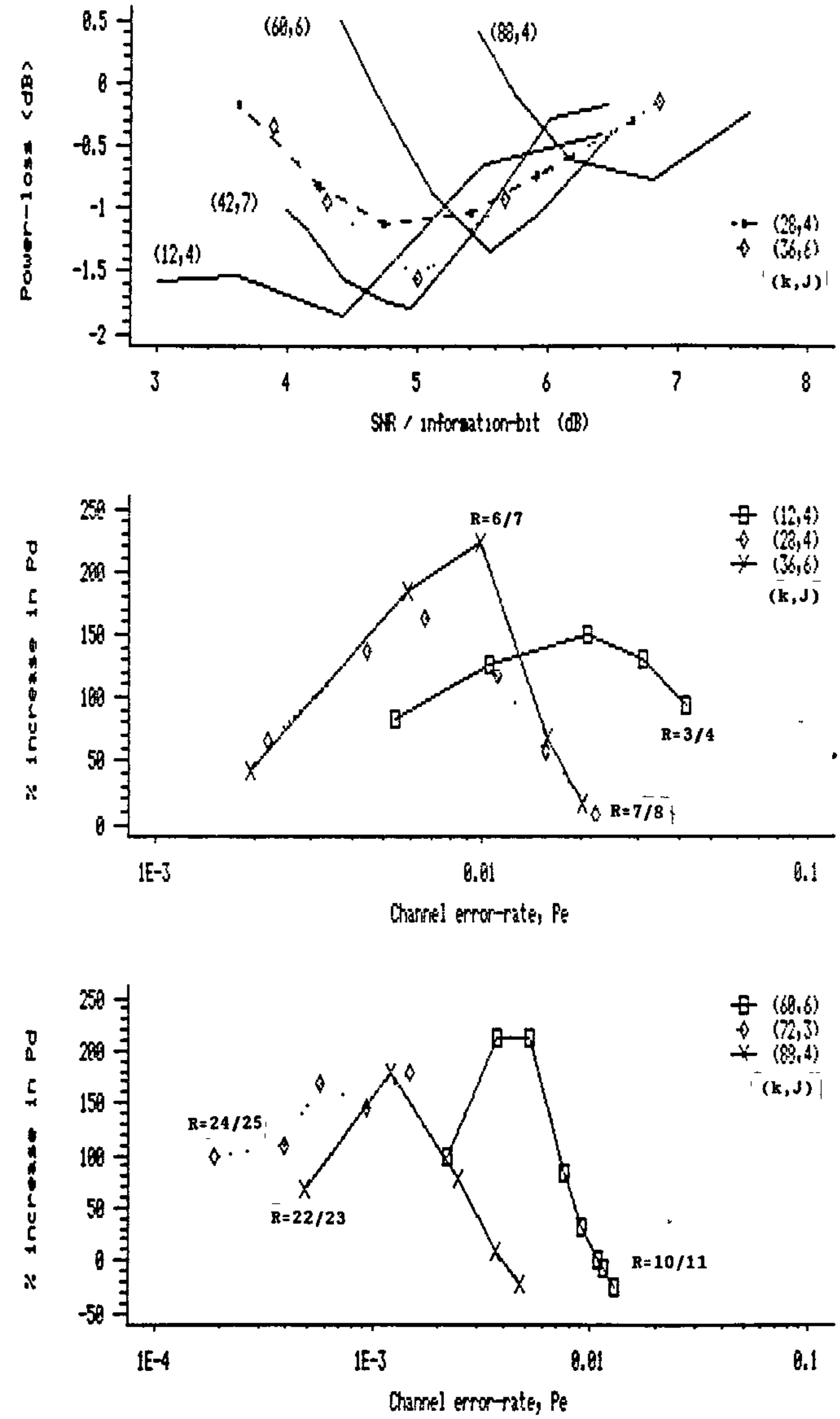


Figure A8.11.6: Power-loss due to error-propagation, vs Γ (top). % increase in decoding errors due to error-propagation vs P_e (middle & bottom). *

* Γ = signal-to-noise power-ratio per information-bit.

APPENDIX 8.12: UNEQUAL ERROR-PROTECTION

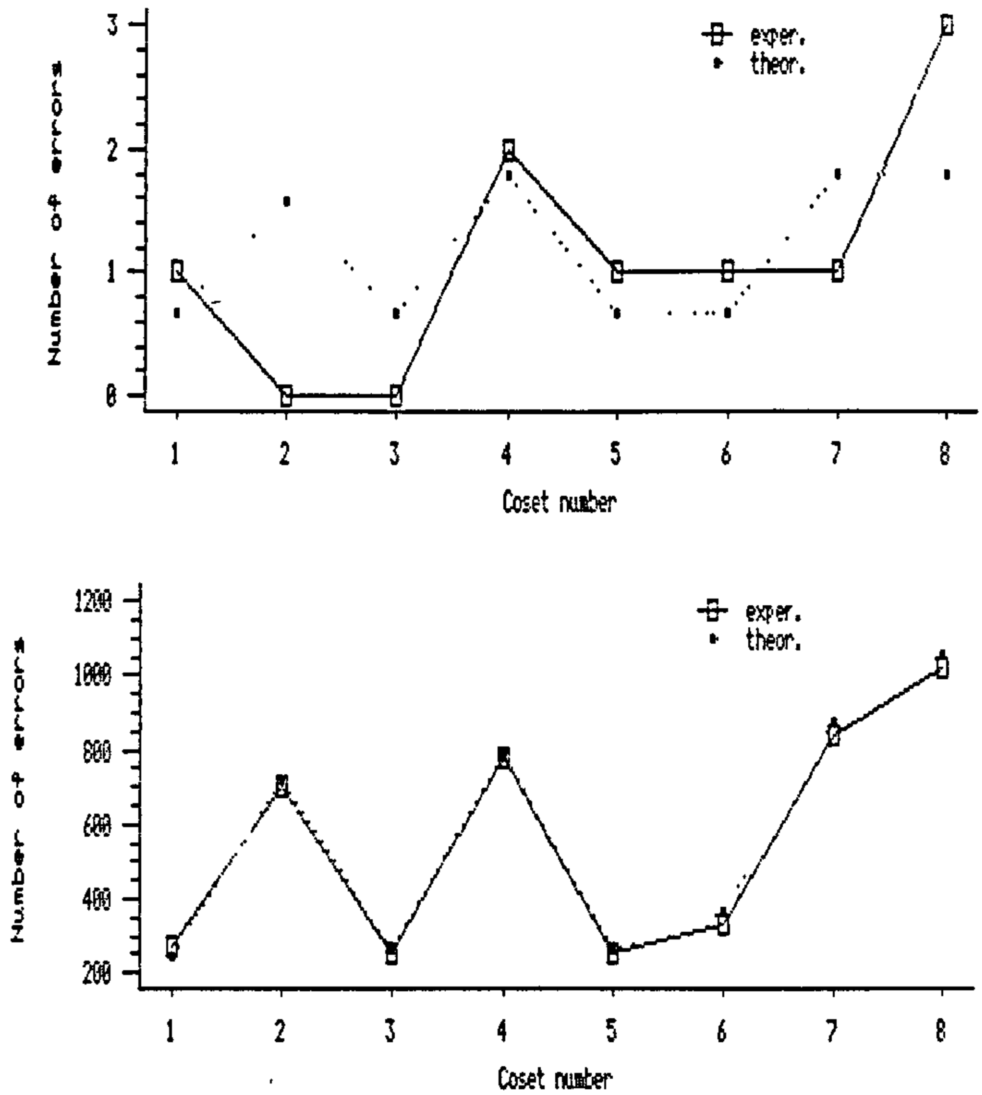
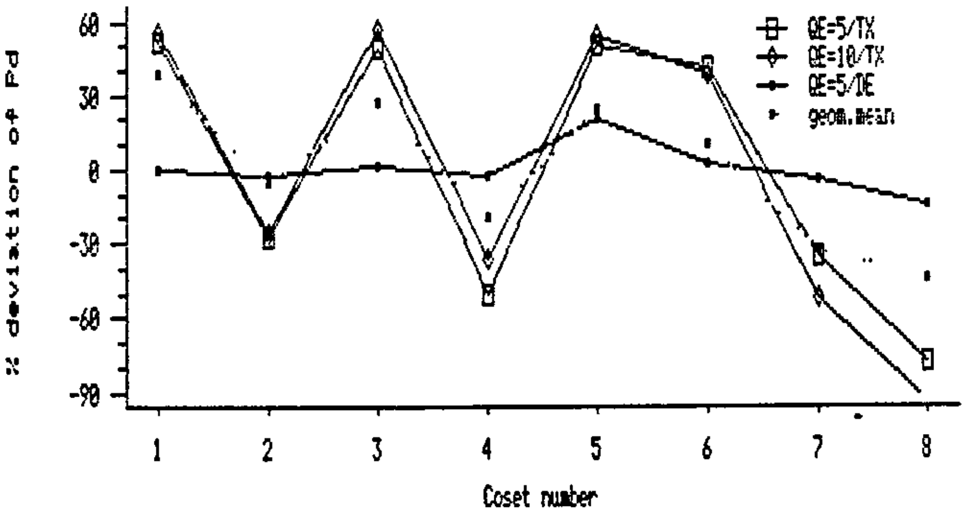


Figure A8.12.1: Number of errors per coset with TX f/b, for the (40,5) code; QE=1 over 45,000 blocks (top) and QE=10 over 20,000 blocks (bottom).



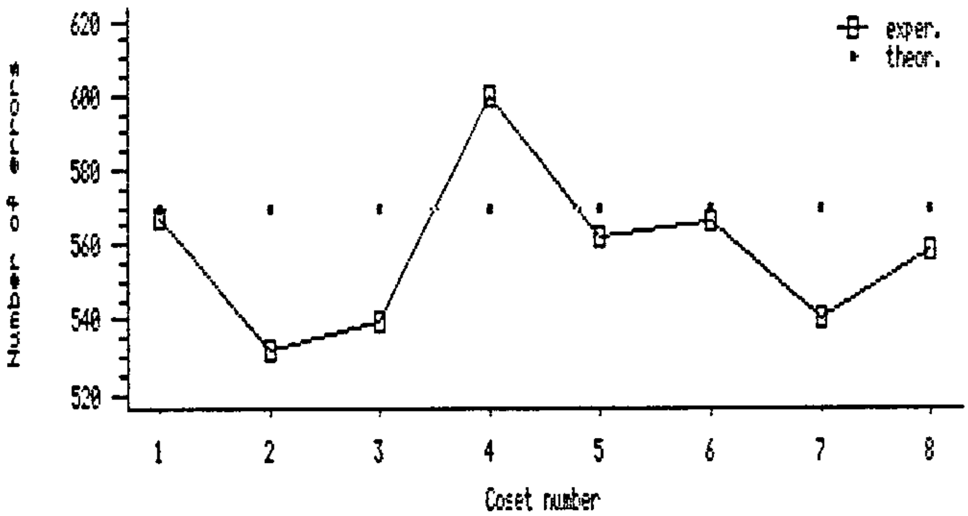


Figure A8.12.2: For the (40,5) code: % deviation of P_d from the average, per coset (top); number of decoding errors per coset, over 20,000 blocks, under DD, with QE=5 (bottom).

References

1. *B. Sklar*: "A Structured Overview of Digital Communications - a Tutorial Review - Part I", IEEE Communications Magazine, pp. 4-17, Aug 1983, and "..... - Part II", pp. 6-21, Oct 1983.

2. *S. Lin & D.J. Costello, Jr*: "Error Control Coding: Fundamentals and Applications", Prentice-Hall, 1983.

3. *A. Papoulis*: "Probability, Random Variables, and Stochastic Processes", McGraw-Hill Kogakusha Ltd, 1965.

4. *W.B. Davenport, Jr*: "Probability and Random Processes", McGraw-Hill, 1970.

5. *W. Feller*: "An Introduction to Probability Theory and its Applications", vol. 1, 3rd edition, John Wiley, 1968.

6. *T.L. Booth*: "Sequential Machines and Automata Theory", John Wiley, 1968.

7. *B. Noble & J.W. Daniel*: "Applied Linear Algebra", Prentice-Hall, 1977.

8. *H.G. Campbell*: "An Introduction to Matrices, Vectors and Linear Programming", Prentice-Hall, 1977, 2nd edn.

9. *F. Ayres, Jr*: "Theory and Problems of Matrices", Schaum's Outline Series, McGraw-Hill, 1974.

10. *R.E. Blahut*: "Theory and Practice of Error Control Codes", Addison-Wesley, 1983.

11. *L.R. Rabiner & B. Gold*: "Theory and Applications of Digital Signal Processing", Prentice-Hall, 1975.

12. *S. Lin*: "An Introduction to Error-Correcting Codes", Prentice-Hall, 1970.

13. *G.C. Clark, Jr & J.B. Cain*: "Error-Correction Coding for Digital Communications", Plenum Press, 1981.

14. *D. Wiggert*: "Error-Control Coding and Applications", Artech House, 1978.

15. *W.W. Peterson & E.J. Weldon, Jr*: "Error-Correcting Codes", 2nd edition, MIT Press, 1972.

16. *R.W. Lucky, J. Salz & E.J. Weldon, Jr*: "Principles of Data Communication", McGraw-Hill, 1968.

17. *J.A. Heller*: "Feedback Decoding of Convolutional Codes", in A.J. Viterbi (ed), "Advances in Communication Systems", vol. 4, Academic Press, pp. 261-278, 1975.

18. *J.L. Massey*: "Threshold Decoding", MIT Press, 1963.

19. *J.L. Massey*: "Advances in Threshold Decoding", in A.V. Balakrishnan (ed), "Advances in Communication Systems", vol. 3, Academic Press, pp. 91-115, 1968.

20. *G.D. Forney, Jr*: "Convolutional Codes I: Algebraic Structure", IEEE Trans. Inf. Theory, vol. IT-16, pp. 720-38, Nov 1970.

21. *G.D. Forney, Jr*: "Structural Analysis of Convolutional Codes via Dual Codes", IEEE Trans. Inf. Theory, vol. IT-19, pp. 512-8, Jul 1973.

22. *J.L. Massey & M.K. Sain*: "Inverses of Linear Sequential Circuits", IEEE Trans. Comput., vol. C-17, pp. 330-7, Apr 1968.

23. *D.A. Huffman*: "The Synthesis of Linear Sequential Coding Networks", in W.H. Kautz (ed), "Linear Sequential Switching Circuits", Holden-Day, pp. 1-19, 1965.

24. *I.S. Reed & T.K. Truong*: "Error-trellis Syndrome Decoding Techniques for Convolutional Codes", IEE Proceedings, vol. 132, Pt F, pp. 77-83, Apr 1985.

25. *G.D. Forney, Jr*: "Convolutional Codes II: Maximum Likelihood Decoding", Inform. Control, vol. 25, pp. 222-66, Jul 1974.

26. *A.J. Viterbi & J.K. Omura*: "Principles of Digital Communication and Coding", McGraw-Hill, 1979.

27. *J.G. Proakis*: "Digital Communictions", McGraw-Hill, 1983.

28. *A.J. Viterbi*: "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", IEEE Trans. Inf. Theory, vol. IT-13, pp. 260-9, Apr 1967.

29. *J.K. Omura*: "On the Viterbi Decoding Algorithm", IEEE Trans. Inf. Theory, vol. IT-15, pp. 177-9, Jan 1969.

30. *J.M. Wozencraft & B. Reiffen*: "Sequential Decoding", MIT Press, 1961.

31. *R.M. Fano*: "A Heuristic Discussion of Probabilistic Decoding", IEEE Trans. Inf. Theory, vol. IT-9, pp. 64- 74, Apr 1963.

32. *K.Sh. Zigangirov*: "Some Sequential Decoding Procedures", Problemy Peredachi Informatsii, vol. 2, No 4, pp. 13-25, 1966 (engl. transl. in "Problems of Information Transmission", vol. 2, No 4, pp. 1-10).

33. *F. Jelinek*: "Fast Sequential Decoding Algorithm Using a Stack", IBM J. Res. Develop., pp. 675-8, Nov 1969.

34. *H.B. Enderton*: "Elements of Set Theory", Academic Press, 1977.

35. *S. Lipschutz*: "Discrete Mathematics", Schaum's Outline Series in Mathematics, McGraw-Hill, 1976.

36. *N.L. Biggs*: "Discrete Mathematics", Clarendon Press, 1985.

37. *J.L. Massey & R.W. Liu*: "Application of Lyapunov's Direct Method to the Error-Propagation Effect in Convolutional Codes", IEEE Trans. Inf. Theory, vol. IT-10, pp. 248-50, Jul 1964.

38. *J.P. Robinson & A.J. Bernstein*: "A Class of Binary Recurrent Codes with Limited Error Propagation", IEEE Trans. Inf. Theory, vol. IT-13, pp. 106-13, Jan 1967.

39. *J.P. Robinson*: "Error Propagation and Definite Decoding of Convolutional Codes", IEEE Trans. Inf. Theory, vol. IT-14, pp. 121-8, Jan 1968.

40. *E. Kreyszig*: "Advanced Engineering Mathematics", 3rd edition, John Wiley, 1972.

41. *S. Barnard & J.M. Child*: "Higher Algebra", Mackillan & Co Ltd, 1949.

42. *D. McQuilton*: "Some New Results on Majority-Logic Codes for Correction of Random Errors", Ph.D. thesis, University of Loughborough, 1979.

43. *D. McQuilton*: "More High-Rate 'Cyclic' Convolutional Self-Orthogonal Codes", IEE Proc., vol. 127, Pt F, pp. 427-9, Dec 1980.

44. *T.M. Apostol*: "Introduction to Analytic Number Theory", Springer-Verlag, 1976.

45. *W.W. Wu*: "New Convolutional Codes - Part III", IEEE Trans. Communications, vol. COM-24, pp. 946-55, Sep 1976.

46. *I.M. Vinogradov*: "Elements of Number Theory", Dover Publications Inc, 1954.

47. *D. McQuilton*: "Effective Constraint Length of 'Cyclic' Convolutional Self-Orthogonal Codes", IEE Proc., vol. 128, Pt F, pp. 69-73, Apr 1981.

48. *H. Davenport*: "Multiplicative Number Theory", 2nd edition, Springer-Verlag, 1980.

49. *B.C. Erricker*: "Advanced General Statistics", Hodder & Stoughton, 1971.

The thesis was typed by the *author*

The following were used:

Computer: *Amstrad PC1512 HD20*

Printer: *Epson LQ-850*

Wordprocessor: *Wordstar 5*

Graphics: *GEM Draw Plus*

Plotting: *Statgraphics 3.0*

The following statistics were obtained:

Character count (total): 2,717,000 bytes

Word count: 121,751 words

GEM Draw Plus diagrams: 43 diagrams - 251,070 bytes

Statgraphics graphs: 91 graphs - 1,680,192 bytes