ENHANCED CODING, CLOCK RECOVERY AND DETECTION FOR A MAGNETIC CREDIT CARD

by

DANIEL FELIX SMITH

A thesis submitted to the University of Plymouth
in partial fulfillment for the degree of

DOCTOR OF PHILOSOPHY

Centre for Research in Information Storage Technology
School of Electronic, Communication and Electrical Engineering
Faculty of Technology

May 1998

Enhanced Coding, Clock Recovery and Detection for a Magnetic Credit Card

Daniel Felix Smith

Abstract

This thesis describes the background, investigation and construction of a system for storing data on the magnetic stripe of a standard three-inch plastic credit card. Investigation shows that the information storage limit within a 3.375 in by 0.11 in rectangle of the stripe is bounded to about 20 kBytes. Practical issues limit the data storage to around 300 Bytes with a low raw error rate: a four-fold density increase over the standard. Removal of the timing jitter (that is probably caused by the magnetic medium particle size) would increase the limit to 1500 Bytes with no other system changes. This is enough capacity for either a small digital passport photograph or a digitized signature: making it possible to remove printed versions from the surface of the card.

in: inch

To achieve even these modest gains has required the development of a new variable rate code that is more resilient to timing errors than other codes in its efficiency class. The tabulation of the effects of timing errors required the construction of a new code metric and self-recovering decoders. In addition, a new method of timing recovery, based on the signal 'snatches' has been invented to increase the rapidity with which a Bayesian decoder can track the changing velocity of a hand-swiped card. The timing recovery and Bayesian detector have been integrated into one computation (software) unit that is self-contained and can decode a general class of (d, k) constrained codes. Additionally, the unit has a signal truncation mechanism to alleviate some of the effects of non-linear distortion that are present when a magnetic card is read with a magneto-resistive magnetic sensor that has been driven beyond its bias magnetization.

While the storage density is low and the total storage capacity is meagre in comparison with contemporary storage devices, the high density card may still have a niche role to play in society. Nevertheless, in the face of the Smart card its long term outlook is uncertain. However, several areas of coding and detection under short-duration extreme conditions have brought new decoding methods to light. The scope of these methods is not limited just to the credit card.

Contents

Appendices

List of Figures

List of Tables

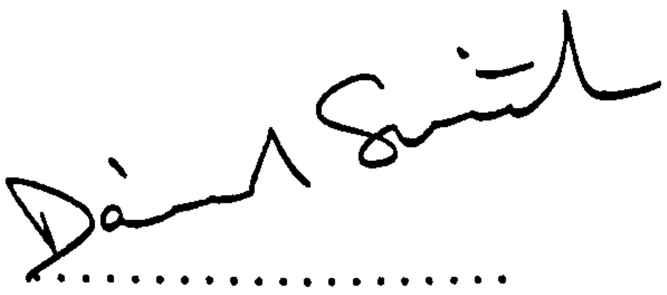
Acknowledgement

This work was funded by the University of Plymouth. Some travel assistance was received from the Royal Academy of Engineering. Thanks are given to my supervisors, Dr. Terry Donnelly and Prof. Desmond J. Mapps, Dr. Paul Davey for maintaining paper references, Nick Fry for valuable assistance, other members of the CRIST facility and everyone else who has provided comment, ideas and support.

Author's declaration

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award.

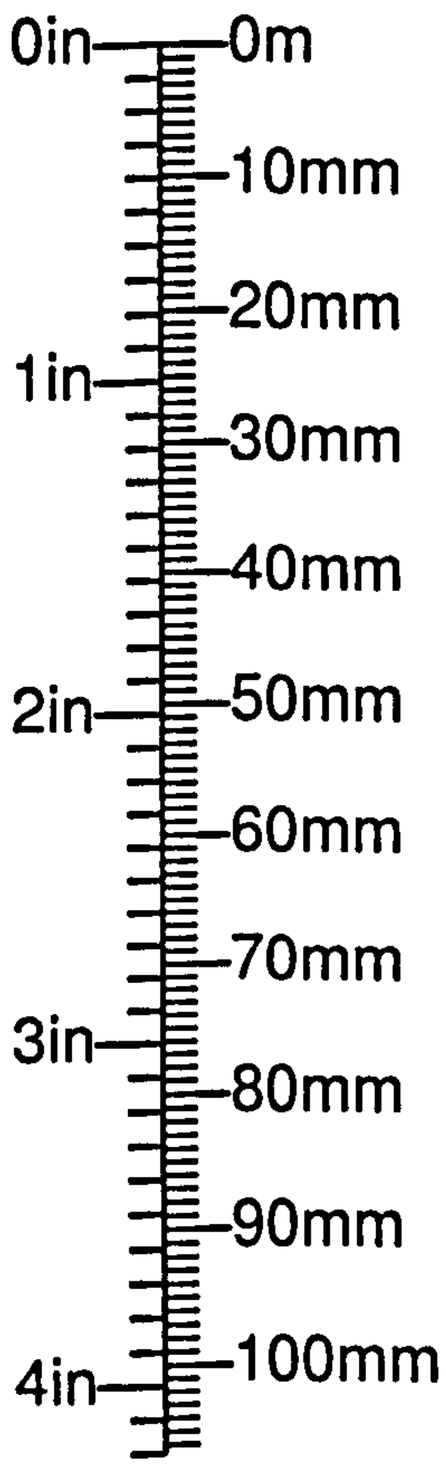
Several scientific seminars and conferences were attended. Relevant publications are reproduced in appendix A. In addition a poster was presented (in absence) to the IEE in London for the colloquium 'Document Image Processing for Multimedia Environment' in October 1995. Four IEEE conferences were attended: Intermag 1995 (San Antonio, Texas), 1996 (Seattle, Washington) and 1997 (New Orleans, Louisiana) and the Globecom conference in 1996 (London, UK). Papers were presented at Intermag in 1996 and 1997.

Signed  Date 29 MAY 1998

Preface

A note on units....

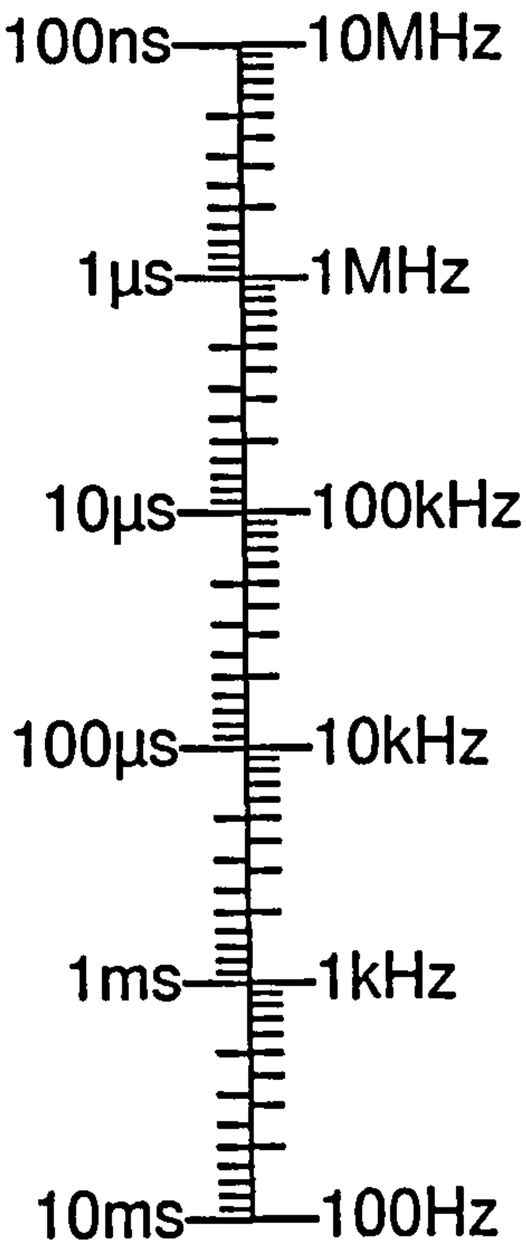
Although the metric system has been in use for over a hundred years, old habits die hard—especially in 'traditional' industries like magnetic recording and banking. The units used in the text of this thesis are all 'common' (i.e., inches for American standards and metric for European/Asian standards). The following charts may be helpful when converting.



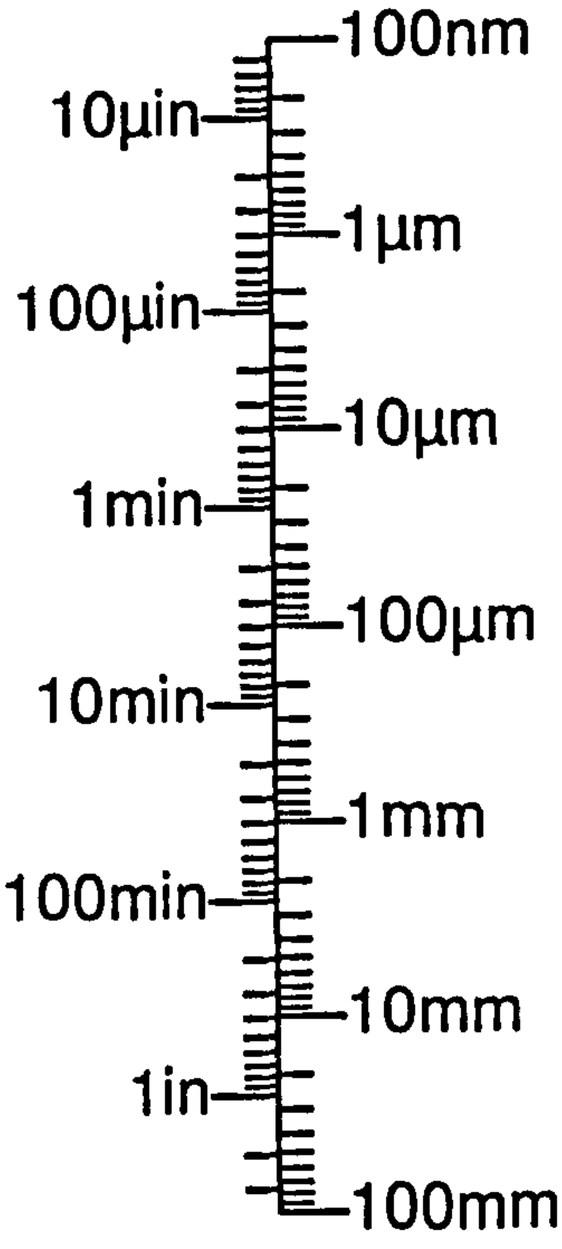
$4\pi \times 10^{-3}$ Oe (oersted) = 1 A/m (ampere/metre)

10^4 gauss = 1 T (tesla)

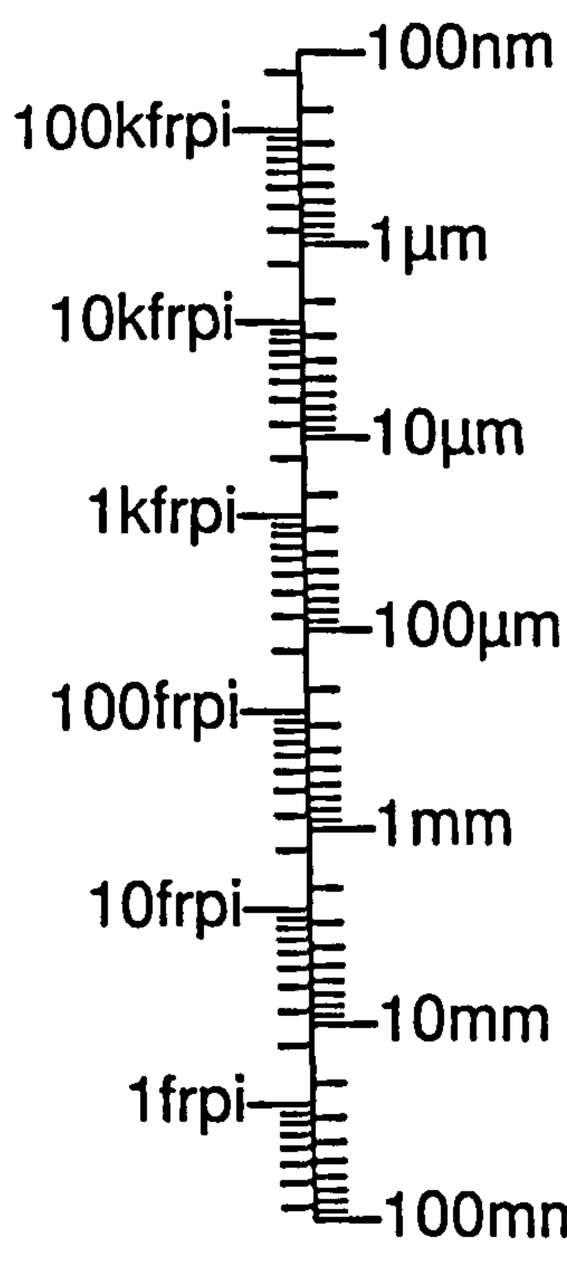
Conversions from (a) time: seconds to hertz (b) length: inches to metres (c) linear density: flux reversals per inch to wavelength (d) areal density: bits per square inch to bits per square millimetre.



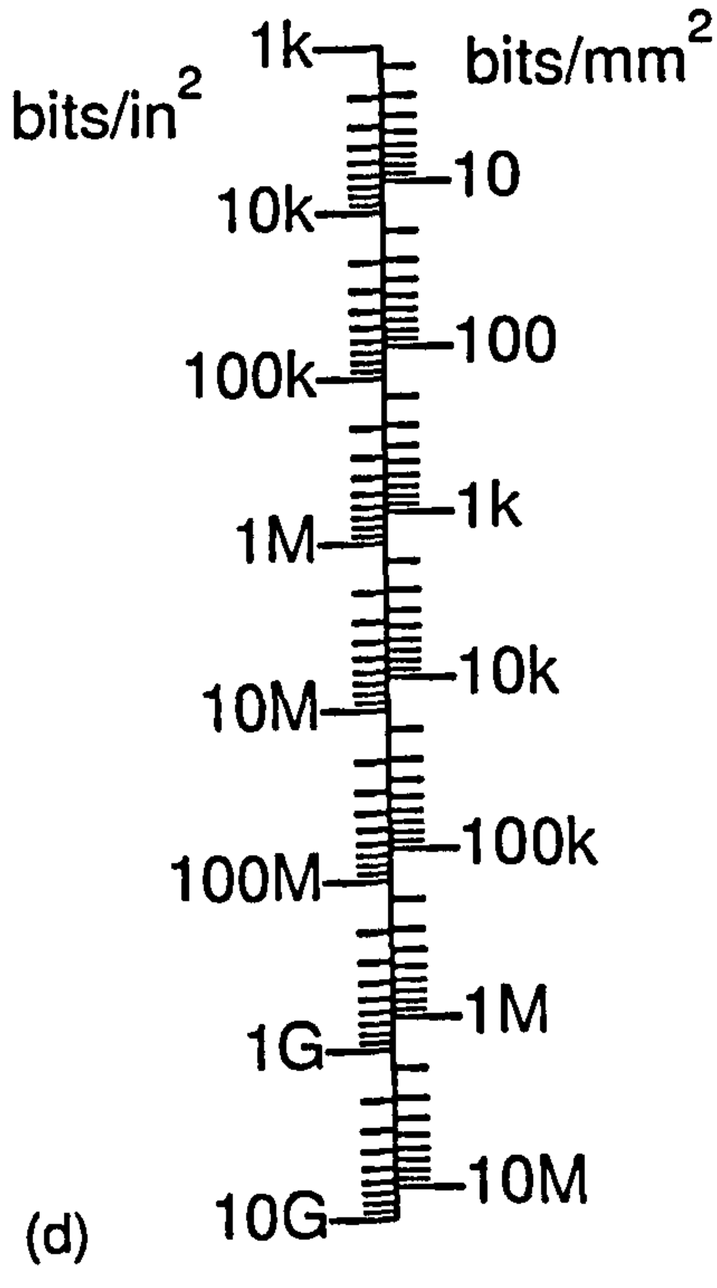
(a)



(b)



(c)



(d)

The following conventions will be used for mathematical notation.

Continuous time is represented by the variable t, and frequency by f. The frequency domain representation of a time domain signal is represented by changing the function's letter to upper case, $g(t) \Longleftrightarrow G(f)$.

A *signal*, say $r(t)$, is a real, analytic, piecewise linear and single valued function. Normally its amplitude is bounded.

When a signal is sampled, the discrete time is written as a subscript, i.e., $r_k = r(kT)$ where T is the sampling interval and k is integer. The letters d, i, j, k, l, m and n represent integer numbers.

A group of samples can be written in vector form: $\boldsymbol{r} = (r_0, r_1, r_2, \ldots, r_{n-1})^T$. In this context a superscript T denotes the transpose of the vector. A matrix is represented by a bold upper case letter, e.g.,

$$\mathbf{M} = \begin{pmatrix} M_{0,0} & M_{0,1} & \cdots & M_{0,n-1} \\ M_{1,0} & M_{1,1} & \cdots & M_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ M_{m-1,0} & M_{m-1,1} & \cdots & M_{m-1,n-1} \end{pmatrix}.$$

The convolution operation is an asterisk.

$$g(t) * h(t) = \int_\tau g(t-\tau)h(\tau)\,d\tau = h(t) * g(t)$$

$$(\boldsymbol{g} * \boldsymbol{h})_i = \sum_j g_i h_{i-j} = (\boldsymbol{h} * \boldsymbol{g})_i$$

The mean operator is denoted by angle brackets.

$$\langle g(t) \rangle_t = \frac{1}{b-a} \int_a^b g(t)\,dt$$

$$\langle \boldsymbol{g} \rangle = \frac{1}{n} \sum_{i=0}^{n-1} g_i$$

The expectation operator is written as a function, E.

$$E(g(t)) = \frac{1}{\langle g(t) \rangle_t} \int_t t g(t)\,dt \qquad E(\boldsymbol{g}) = \frac{1}{\langle \boldsymbol{g} \rangle} \sum_i i g_i$$

$$E(g) = \int_x x \,\mathrm{pr}(g = x)\,dx \qquad E(g) = \sum_i i \,\mathrm{pr}(g = i)$$

The symbol $\mathrm{pr}(\cdot)$ is probability (there is no notational distinction between a continuous density function and discrete). Conditional probability is written $\mathrm{pr}(A|B)$ which reads as 'the probability of A given that B is certain.'

A hat over a symbol, e.g., $\hat{g}$, means that $\hat{g}$ is an estimator of a true g. Although $\hat{g} \neq g$ it is likely that $\hat{g} \approx g$.

Some *real* physical functions can not necessarily be represented or modeled by mathematical expressions, and are written in calligraphic script. However they may be approximated by, say, a convolution. E.g., $\mathcal{H}(w(t)) \approx h(t) * w(t)$.

Chapter 1

Introduction

Credit cards are durable, inexpensive, machine readable, reliable plastic rectangles. They are not secure, high capacity or indestructible. This project investigated increasing the data storage capacity on a standard magnetic credit card stripe while not significantly altering any of its other features.

Certainly the magnetic stripe card has direct competition in its market arena, but despite fierce onslaught the simple painted magnetic storage medium has held its ground, and is likely to do so for at least ten more years.

A consequence of the success of the format is that almost no development of the medium has occurred since its inception in the 1960s and 70s for fear of disrupting the de facto standards. The technology of other forms of magnetic and communication device has undergone continual improvement in this time so that current storage densities, costs and transmission rates are several orders of magnitude beyond their twenty-year-old counterparts.

The cause of these rapid advances can be traced to refinement in materials, manufacturing processes and the availability of computing power. The cause of stagnation in the credit card is the miniscule cost of the medium itself, its simplicity and the installed base of card readers and infrastructure.

However, it is still desirable to increase the storage capacity of the magnetic credit card. Any high density magnetic card system should, at the very least, maintain low costs, but it is also advantageous to remain to some extent backwards compatible with the current card standard. The extra information should

be an *extension* to a card's facilities, not a replacement for them. Additionally, the card readers should neither be prohibitively expensive nor too stringent on read-back conditions.

This chapter starts with a basic examination of magnetic recording, which is followed by a description of the procedure and formats used on current credit cards. There is an acknowledgement of the competition that is competing for the magnetic card market, and a section on the uses of token cards in general. Errors and their correction are also covered in this chapter, which finishes with brief statements of the contribution and organization of this thesis.

1.1 Fundamentals of magnetic recording

The limit to data capacity on a card is tied to the physics of magnetic recording. Practical magnetic recording began with audio reproduction on steel wire [12]. The state of the art is digital recording on thin-film keepered media [41] on a super-smooth glass disk substrate [84]. Magnetic detectors can be as simple as a loop of wire or Hall-effect semiconductor to sophisticated multi-layer quantum effect devices. Somewhere in the middle of this technology lies the everyday magnetics that is straightforward and robust enough to be useful on a credit card.

This section describes simplified practical magnetic recording. The full theoretical treatment of magnetic recording is intricate and is of limited usefulness. A revision of magnetic fields comes first, then a qualitative model of recording on magnetic media is presented and that is followed by the methods of recording, including a look at magnetic recording and replay heads and practical issues associated with them. There is a brief statement of the limits that a head places on the channel capacity. Finally the relationship between the magnetic recording channel and information capacity is examined. The section is meant to convey the limitations of magnetic recording and some of the underlying principles that will later be used when modeling the channel—most importantly that of magnetic field superposition. It does not cover the physical

processes or precise micromagnetic simulations.

1.1.1 Magnetic fields

The phenomenon of magnetism is the manifestation of four-space electromagnetism being twisted by special relativity. Fortunately, at ordinary velocities Maxwell's equations (see any relevant physics text book, e.g., [37]) describe the interactions between electric and magnetic fields comprehensively. Even better, most magnetic recording is concerned only with magnetostatics, superposition and induced voltage.

A magnetic dipole moment, $\boldsymbol{m}$, is defined in terms of a current, I, flowing around a tiny loop, $\boldsymbol{s}$. This vector $\boldsymbol{s}$ has the properties that its orientation is normal to the loop's area and its magnitude is equal to the loop's area. The relationship in SI units between the magnetic dipole and electric current is

SI: Système International

$$\boldsymbol{m} = I\boldsymbol{s}. \tag{1.1}$$

$\boldsymbol{b}$ = *magnetic field*
$\boldsymbol{r}$ = *position vector*
$\boldsymbol{m}$ = *dipole vector*
$r = |\boldsymbol{r}|$
$\mu_r\mu_0$ = *permeability*

The magnetic field*, $\boldsymbol{b}(\boldsymbol{r})$, at position $\boldsymbol{r}$ relative to the dipole is related to $\boldsymbol{m}$ by

$$\boldsymbol{b}(\boldsymbol{r}) = \frac{\mu_r\mu_0}{4\pi} \cdot \nabla\left(\frac{\boldsymbol{m}\cdot\boldsymbol{r}}{r^3}\right) \tag{1.2}$$

where $\frac{\mu_r\mu_0}{4\pi}$ (the material's relative permeability and permeability constant) are treated as constant and $r = |\boldsymbol{r}|$, the magnitude of $\boldsymbol{r}$.

Magnetic fields are linear and superposable, so equation (1.2) essentially says that the prevalent magnetic field is the sum of the contributions from all nearby magnetic dipole moments, which reduce in intensity at r^{-3}.

Ferromagnetic particles in a magnetic medium behave like many independent closed loop current dipole moments† so some simple magnetic calculations can be made. For a recorded data track with a width much larger than the width

*In this work no distinction is made between B and H fields. The concepts of 'flux density', 'magnetic field' and 'magnetizing force' are all represented by the vague symbol b or $\boldsymbol{b}$.

†It should be noted that the strength of the magnetic dipoles is dependent on the volume of the particle and the material of which it is made. There is also a shape dependence that affects coercivity.

of the read head, the dipoles can be arranged as bars or rods across the track. The field contribution from a bar of dipole drops off as approximately r^{-2}, derived below as equation (1.5).

The exact equation for the contribution to the magnetic field from a bar of m-strength dipole aligned on the z-axis from z_a to z_b as derived from equation (1.2) is

$$b(r) = \frac{\mu_r \mu_0}{4\pi} \cdot \left[\frac{1}{r^3} \mathrm{M}(r) \cdot m \right]_{z_a}^{z_b} \tag{1.3}$$

where $\mathrm{M}(r)$ is a positional scaling factor with physical units of r,

$$\mathrm{M}(r) = \begin{pmatrix} \frac{z\{(x^2+y^2)(-2x^2+y^2)-z^2(x^2-y^2)\}}{(r^2-z^2)^2} & \frac{-xyz(3r^2-z^2)}{(r^2-z^2)^2} & x \\ \frac{-xyz(3r^2-z^2)}{(r^2-z^2)^2} & \frac{z\{(x^2+y^2)(-x^2+2y^2)-z^2(x^2-y^2)\}}{(r^2-z^2)^2} & y \\ x & y & z \end{pmatrix}. \tag{1.4}$$

v^T = *transpose of* v

Here $r = \begin{pmatrix} x & y & z \end{pmatrix}^T$ is the position relative to the dipole bar and $[f(z)]_a^b$ denotes the limits of integration, $f(b) - f(a)$.

For a wide dipole bar (length $\gg$ the width of the head, $-z_a = z_b \to \infty$) and the dipole moment, m, aligned on the y-axis, $m = \begin{pmatrix} 0 & m & 0 \end{pmatrix}^T$, the long expression (1.3) reduces to simpler

$$b(r) = \frac{\mu_r \mu_0}{4\pi} \cdot \frac{4m}{(x^2+y^2)^2} \begin{pmatrix} xy \\ y^2 - x^2 \\ 0 \end{pmatrix}. \tag{1.5}$$

These magnetic vector fields are illustrated in figures 1.1 and 1.2 which show field cross-sections for a point dipole and bar dipole respectively. Figure 1.3 shows the field from a set of small bar dipoles as might be seen by a magnetic sensor over a digitally recorded magnetic medium (the situation is similar to that shown later in figure 1.6). The vertical component of the magnetic field in figure 1.3 at different elevations over the medium is shown in figure 1.4: this demonstrates that there is substantial loss of signal bandwidth and strength as the distance from the magnetic sensor to the medium is increased. The reduc-

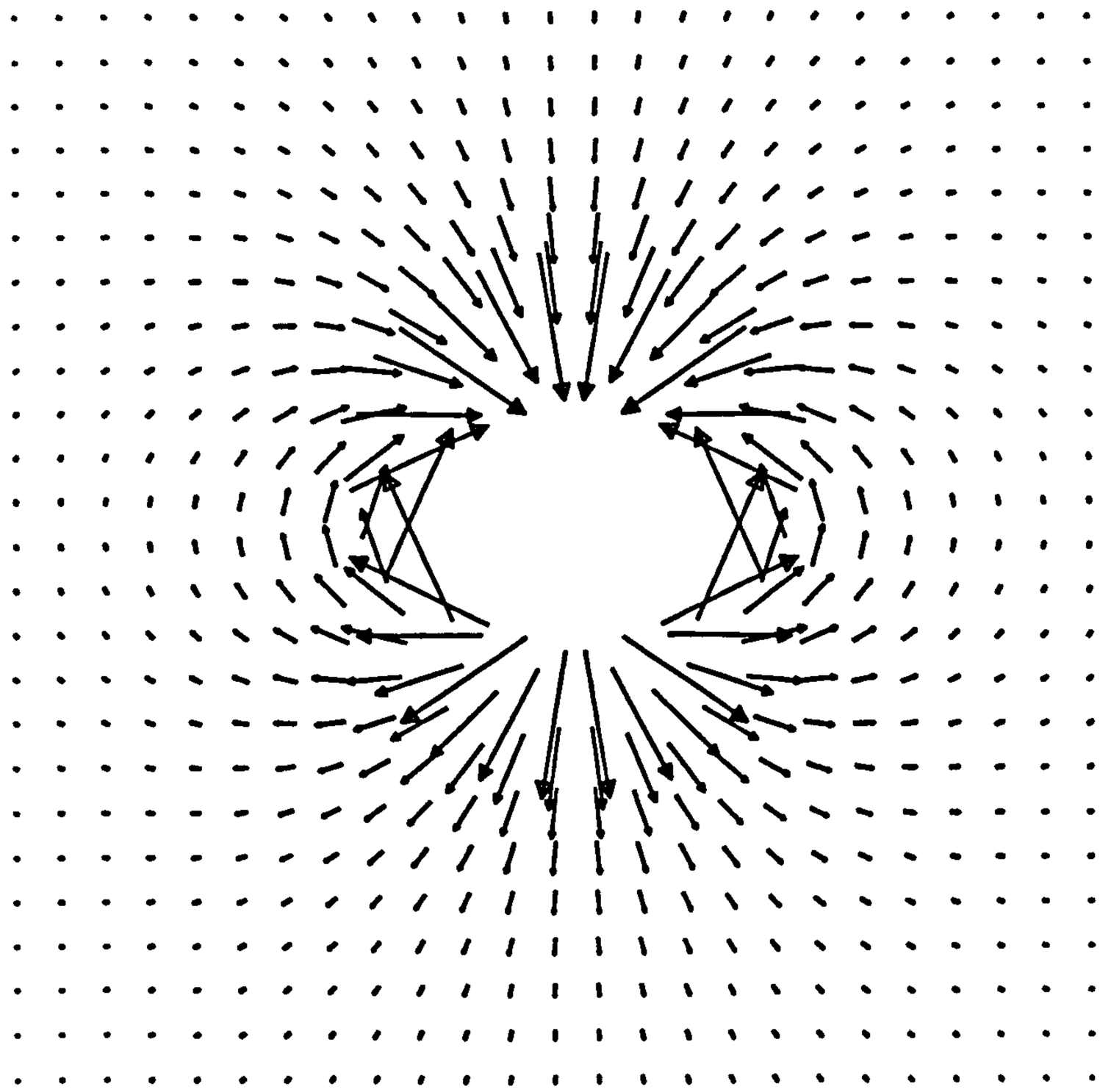


Figure 1.1: Magnetic field cross-section from a point dipole aligned to point down the page. The field drops in intensity at $1/r^3$.

tion of detail is known as Stevenson loss [9] and it is the principal data limiting factor on a magnetic card.

Micromagnetic modeling techniques use the full versions of the above equations to generate the theoretical three-dimensional fields from models of the head and medium. They can include the interaction of dipoles with the writing current (giving rise to high density non-linear distortion), the different permeabilities and orientations of the materials present and even, sometimes, the dynamic effects from Maxwell's equations. This results in a very accurate, very computationally intensive model. (One recent example of thick media modeling is [22]).

Magnetic models are often needed for very long data runs where non-linear

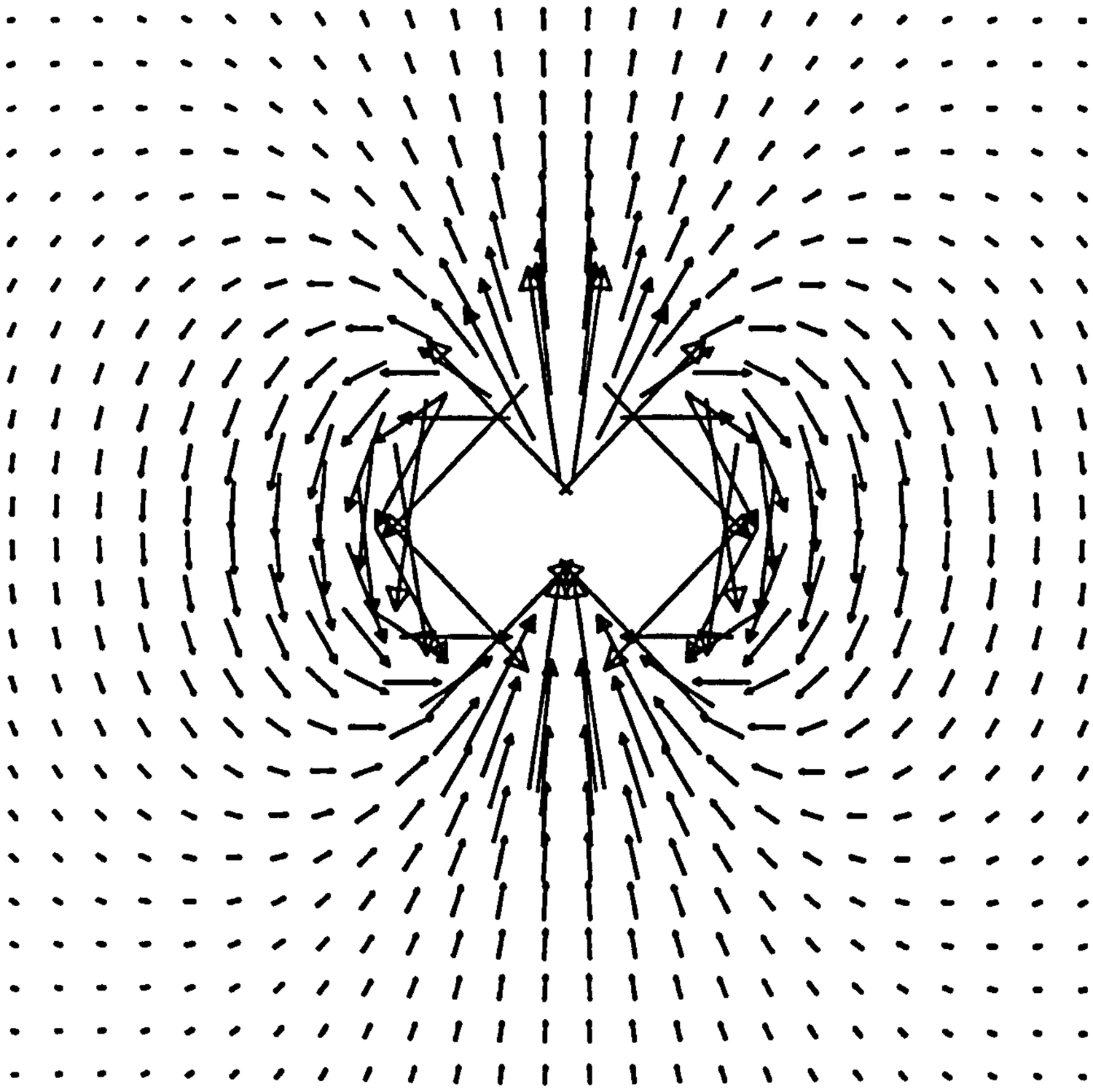


Figure 1.2: Magnetic field cross-section from an infinite bar along the z-axis (going in and out normal to the page) with its dipole moment aligned on the y-axis (vertical, up the page). The field drops in intensity at $1/r^2$.

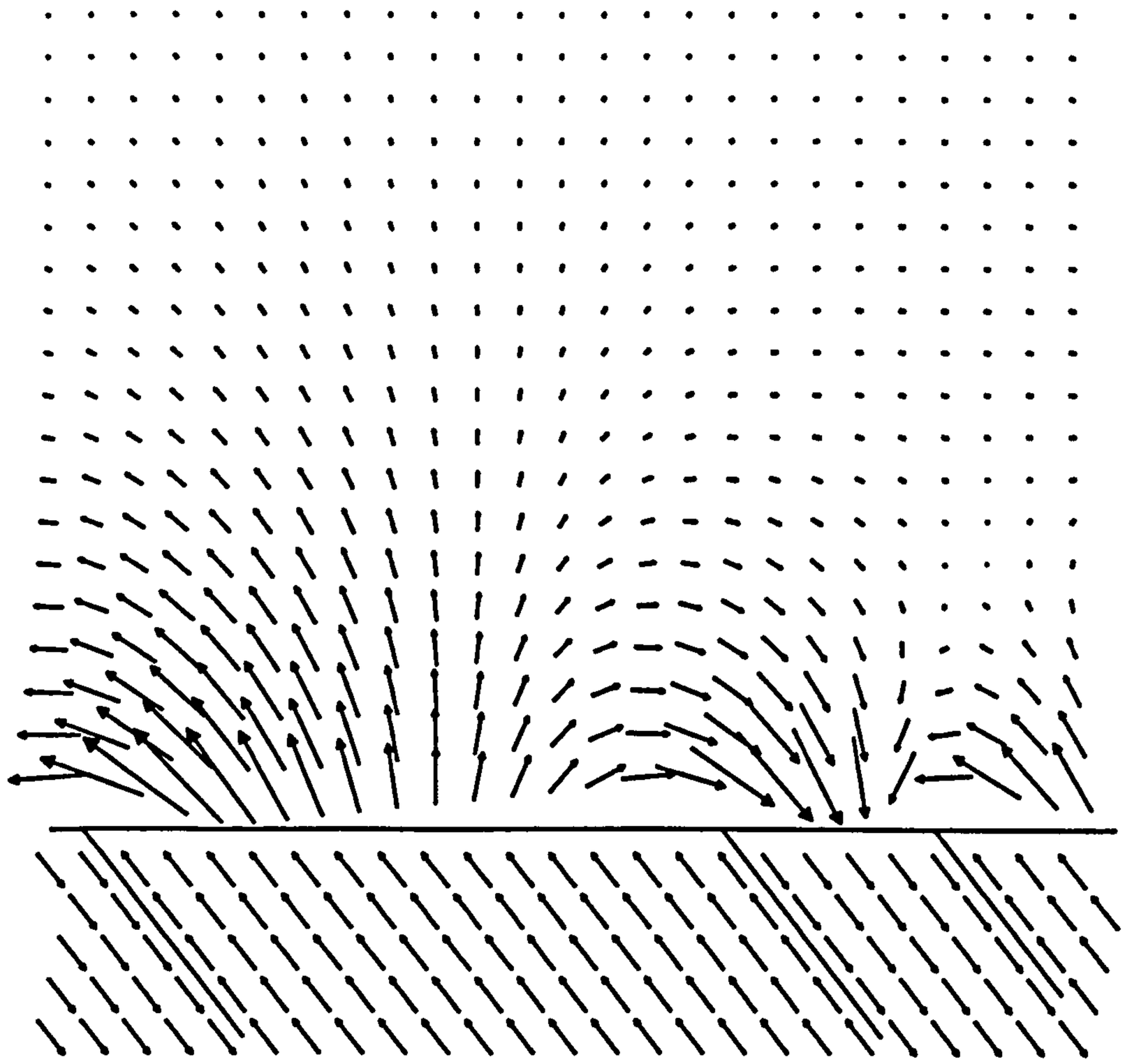


Figure 1.3: Magnetic field cross-section from a set of bar dipoles, similar to a piece of high density recorded media. The (infinitely deep) dipoles are shown in the bottom part of the diagram, the magnetic vector field in the top part. Note how the field extends further out from large areas of magnetization than from smaller ones.

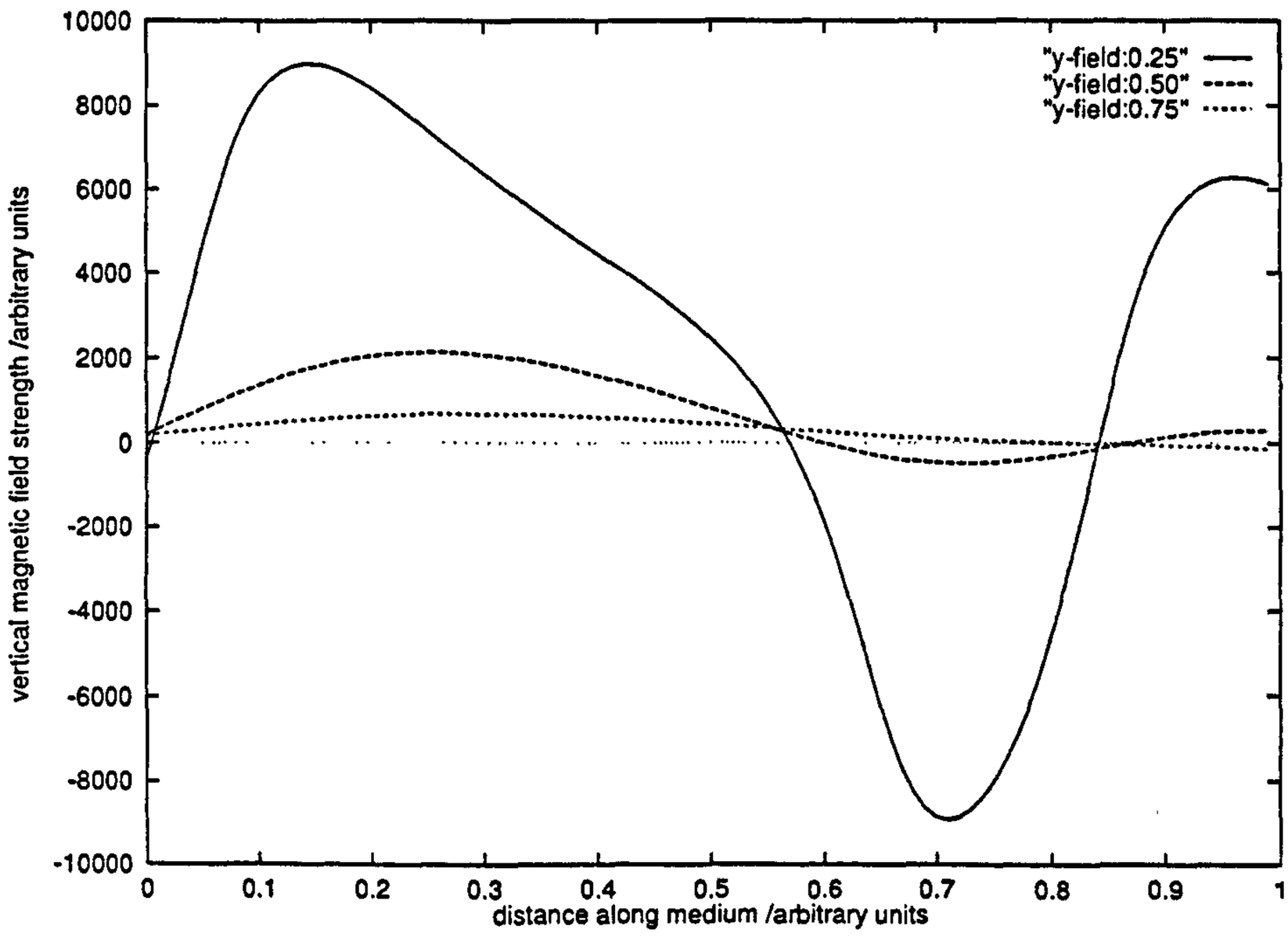


Figure 1.4: Vertical magnetic field component from the set of bar dipoles in figure 1.3. Note the drop in signal detail (bandwidth) and signal strength as the distance to the medium increases. (With relation to figure 1.3, y-field:0.25 is at the medium boundary, y-field:0.50 is at the mid-line of the diagram, y-field:0.75 is mid-way between the top of the diagram and y-field:0.50.)

distortion is in evidence and a micromagnetic model would be too computationally intensive. Simpler semi-empirical systems [52, 53] can provide results almost as good as the full models while still being practical for collecting meaningful long-term channel statistics.

1.1.2 The magnetic recording media as given

A great deal of research has gone into making magnetic recording media that are suitable for high bit densities. However, exotic materials and precision engineering comes at a price while the credit card's most important feature is its low cost.

Taking the standpoint that the credit card medium *cannot* be altered, only the salient and common properties of all media must be examined: not in a theoretical way (the variations in particle size, coercivity and remnance are far too great to make this useful here) but in a purely empirical and phenomenological manner. This has shifted the scope of the magnetic medium into the channel properties; the medium itself has not been explicitly studied.

1.1.3 Trivial micromagnetic hysteretic medium model

The magnetic hysteresis of ferromagnetic recording media is very well documented [50]. A simple (and not totally unrealistic) model is to assume that a ferromagnetic grain has a magnetic dipole moment. When an external magnetic field acting on the grain reaches a critical strength, H_c, the dipole instantly aligns itself with that field.

In this application it was assumed that the recording field strength was always large enough to align every grain, regardless of size and coercivity; i.e., the write process itself does not contribute significantly to the channel characteristics.

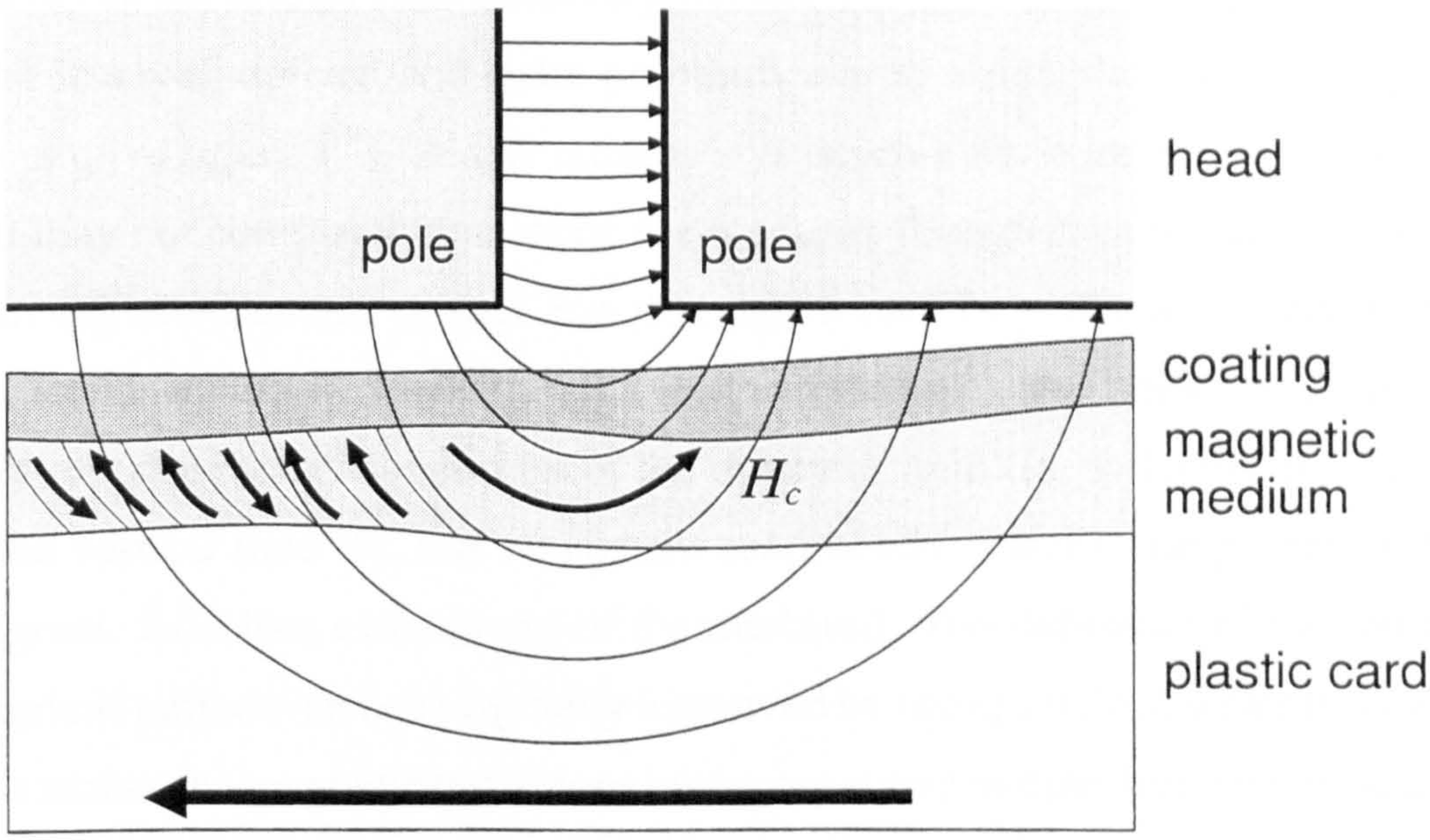


Figure 1.5: Sketch of the magnetic pattern laid down by the trailing edge of the (unperturbed) field of a recording head. If the write gap is very small, the critical field, H_c, does not penetrate down to the magnetic medium.

1.1.4 Magnetic recording (write) heads

All magnetic recording heads rely on an electromagnetic field generated by an electric current flowing through a coil. The flux is ducted (through low reluctance material) to the poles, which are physically near the recording medium. Figure 1.5 is a sketch of the field from a recording head, and shows the flux from the poles being stored in the medium.

The space between the poles, called the head gap, is critical in determining the strength of the magnetic field delivered to the medium [49] if the current is limited or the head is easily saturable: a larger gap increases both the extent and strength of the protruding field. Note, however, that the gap size per se does not influence the maximum recordable linear bit density significantly.

The shape and definition of the trailing edge of the critical write field is affected by the current through the write head. When using a thick, isotropic medium, a high current gives a transition edge that is distant from the gap,

and hence the critical field boundary (where $|b(r)| < H_c$, ignoring units) tends to be less well defined and more perpendicular to the medium (up and down the page in figure 1.5). A low current lays down a more longitudinal pattern that may not completely penetrate the medium. This perpendicularity has not been exploited in this project, for although it may be possible to write to the medium in a variety of orientations, simple read heads are limited to detecting only one directional component of the magnetic field (for inductive heads this is the vertical field [9], but all current read heads detect a component of the magnetic field that extends out of the medium). The definition of the written magnetic pattern can be marginally improved by using a pulsed writing system. This makes the magnetic transitions better separated in time, but not necessarily in space: although the former often compensates for the latter to some extent.

In the credit card application, the dynamic effects in the write head are small enough to ignore. The heads are modestly inductive, but the speed of current change through the head is fairly slow—less than 80 kHz most of the time and the write amplifier is capable of sourcing about an amp of electrical current—so the field rise times are much shorter than the clock period. In systems where the write frequency is above 10 MHz or so, head inductance becomes a serious issue.

1.1.5 Magnetic replay (read) heads

Any magnetic sensor can be the active part of a read head. Currently, inductive coils and magneto-resistors (MR) are the most popular sensors and spin valves (giant magneto-resistors, GMR) have recently been used in products. Figure 1.6 is a sketch of the fields that a read head might see. The flux lines are channeled from the medium through the pole tips to the magnetic field sensor.

The gap size of the read head helps to determine the maximum readable bit density. If the bit size is less than the gap size then several bits can 'fit inside' the gap and interfere with each other. In this application, however, the gap size of the read head is not the limiting factor.

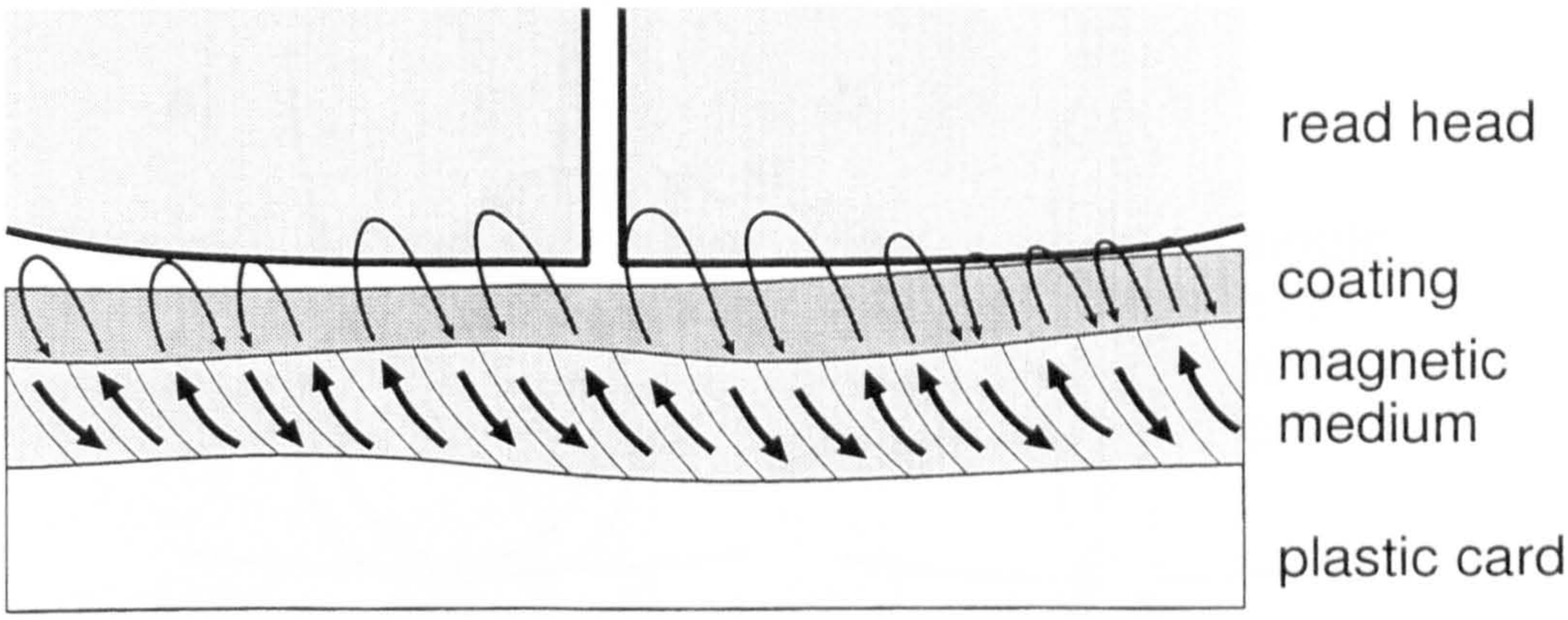


Figure 1.6: Sketch of the (unperturbed) magnetic field seen by a read head. The closer the transitions, the smaller the magnetic field strength seen by the read head.

1.1.6 Limiting factors for signals and heads

The most important constraint limiting the bit density of many recording systems, including this one, is the head to medium separation. This changes as approximately −50 dB per wavelength of recorded signal (which is of the order of the length a bit in the medium). The separation loss is experienced on both the read and write sides of the channel [49] and this loss of detail was displayed in figure 1.4. When the signal is highly attenuated the noise characteristics of the head become important, although in this project noise was not a major issue.

Other, lesser, factors limiting the read-back bandwidth include the read gap size, head reluctance (important for very high frequency recording), azimuth offset (which affects the effective size of the head gap) and head shape [65, 99].

1.1.7 Practical inductive heads and their properties

The two head types examined in this work have been the inductive read/write head and MR read/inductive write head which are shown schematically in figure 1.7. MR heads are described in the section 1.1.8.

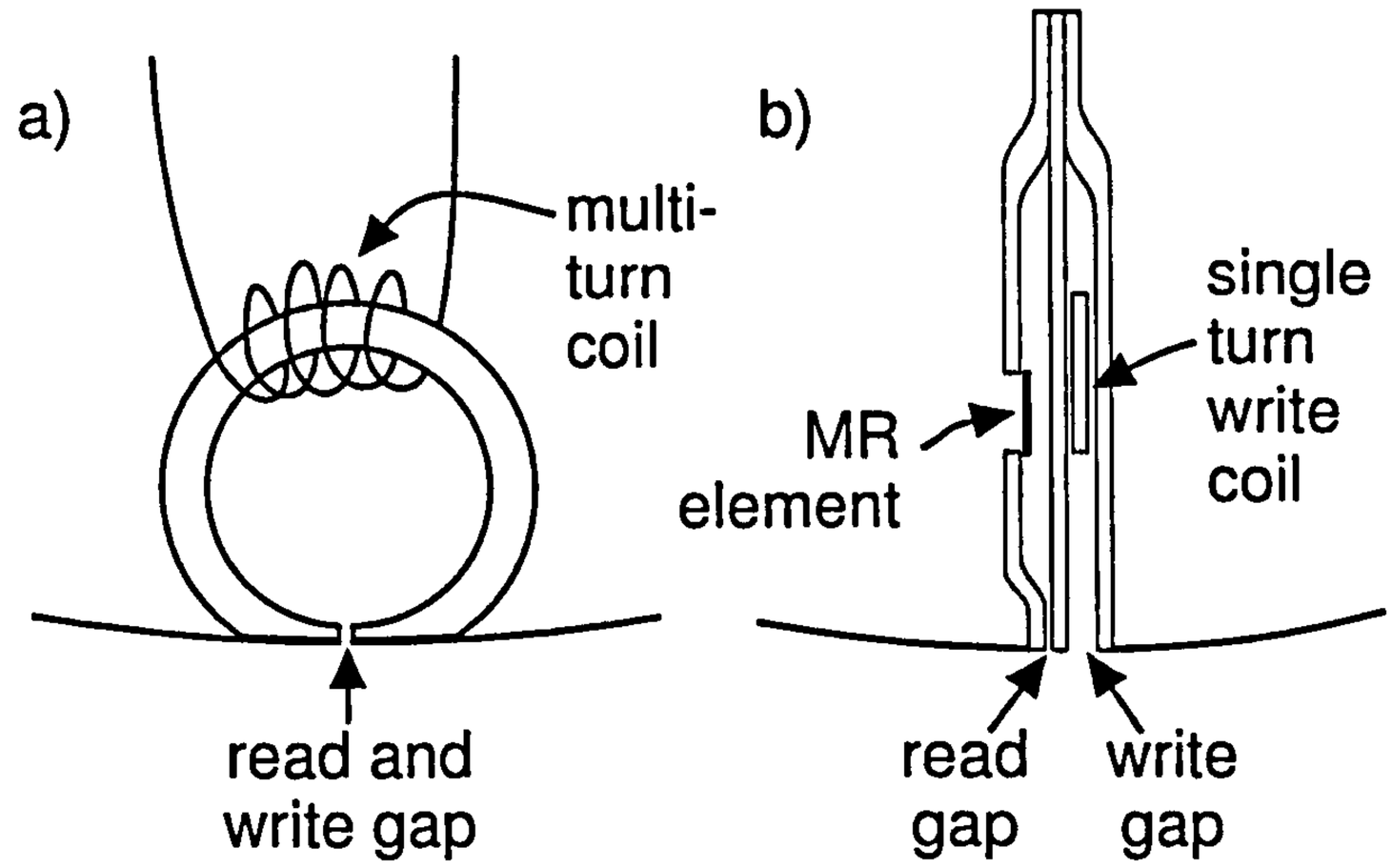


Figure 1.7: Schematic cross-sections of types of writing and reading heads. a) Classic inductive head. b) Thin-film MR read and inductive write head (based on Philips' Digital Compact Cassette head).

Electrical properties of inductive writing

A coil winding is an inductive load, so the number of turns, size, efficiency and relative permeability of the core help determine the voltage required for generating a high frequency magnetic field. For this application the frequencies are mostly less than 10–20 kHz and the voltages are less than about 10 V. With higher frequency heads (several MHz), the inductance can cause serious problems.

The magnetic flux generated by the coil is linear with the coil current (at field strengths less than the core's saturation level) and the number of turns of the coil. With a single turn head, this can mean that very high currents are required, at which point energy absorption becomes important [69]. For this reason, in some applications the head coils are pulsed with a very high instantaneous current at low duty cycles. Using this method means that the extra expense of adding more than one coil turn can often be avoided while still achieving critical magnetic fields and not destroying the head.

Electrical properties of inductive reading

b = magnetic field

Faraday's law, part of the Maxwell equations, states that in a fixed system

$$V \propto \frac{db}{dt} \tag{1.6}$$

EMF: electro-motive force

where t is time and V is the electro-motive force (EMF or voltage) generated across an open loop of conductor with mean magnetic field, b, passing through it. Simply, the faster the field intensity changes, the higher the voltage that is generated.

In storage systems, $\frac{db}{dt}$ is proportional to the medium speed relative to the head. This can be a problem in a variable speed system such as a credit card reader as amplitude variations can affect later stages in decoding, such as adaptive filters and threshold detectors.

1.1.8 Magneto-resistive (MR) heads

MR: magneto-resistor

An MR element has a resistance, R, that is dependent on the magnetic field strength resolved along the element's sensitive axis, b_x; i.e., $R = \beta(b_x)$ [49]. This relationship, $\beta(\cdot)$, is neither linear [64] nor particularly extensive: the change in R is often only a few %.

To detect a change in resistance, a current must be passed through the head. This is used to generate a voltage with the magnetic signal function superimposed on the top of it. Because the useful signal is so small, the voltage from the sense current is removed by a DC blocking capacitor. This modifies the signal at low frequencies, but if the impedance of the input amplifier is high enough, and the (d, k) (and DC) properties of the code are suitable, the modification is not of particular concern.

DC: direct current

Very small MR heads generate more noise than inductive heads. In thin-film head devices this is mostly Barkhausen noise from domain switching and the industry has made considerable efforts to study this and reduce its effects [83]. In a tightly defined recording channel, the noise is the information limiting fac-

tor (see equation (5.2), page 94). At moderate recording densities and high magnetic field strengths, it is less relevant.

1.1.9 Saturation and flip-back in an MR head

An inductive read head is a linear device: the output is linearly proportional to the rate of change of magnetic flux through its coil. A magneto-resistive head is not linear: the output is a function, β, of the magnetic flux through the transducer. Although the MR function is (in places) approximately linear [64] there are exceptional regions at the saturation limits and the zero field point.

The typical form of an MR transfer function is shown in figure 1.8 (the formula $\beta(b_x) = (\tan^{-1}(b_x))^2$ has been used for the purpose of illustration—there are three broad classes of MR transfer function of which the form shown would be classified as type I [18]). The head is usually biased (by an external magnetic field) to a working region at b_b. This is normally near the point of inflection (for best linearity) or the half-way value (for best range of field).

If the medium field becomes large in the same direction as the bias field, the MR transducer saturates gracefully. However, if the medium field overpowers the bias field, then an unfortunate 'flip-back' condition can occur. This signal inversion can confuse a discrimination circuit. Figure 1.9 shows a real example of an MR signal with inversion as read from a credit card.

Avoiding the condition

In closed or well defined systems (disk drives, tape units) this is not a problem: the MR sensor can be chosen to match the medium field, and the bias field can be adjusted as appropriate.

In a credit card environment this solution is not workable. The vague and variable tolerances on the card, medium and surface tribology ensure that signal inversion cannot be avoided without detrimentally affecting the MR sensitivity, although it may be possible to engineer a solution to lessen the impact of flip-back.

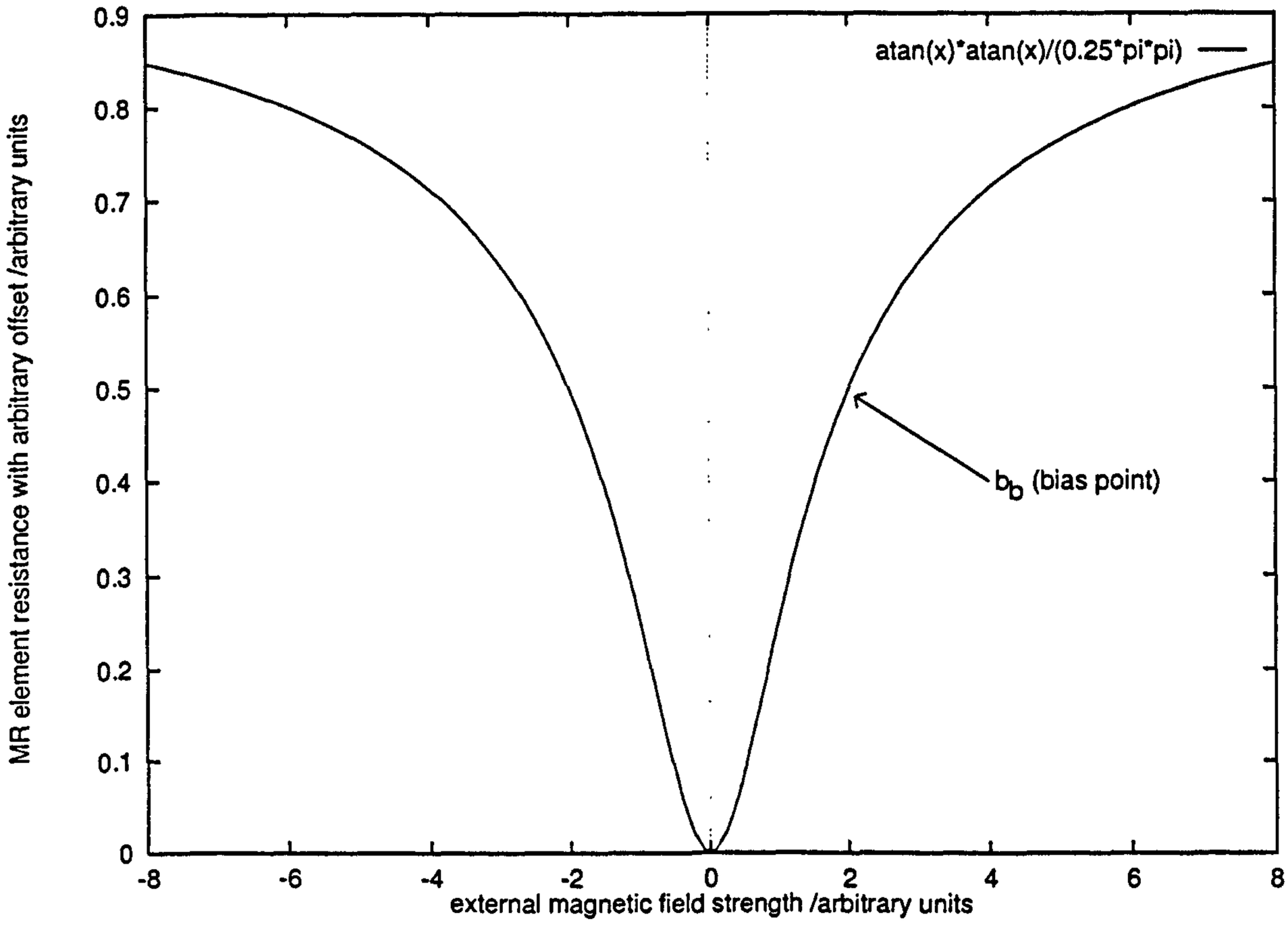


Figure 1.8: Typical form of MR element transfer function.

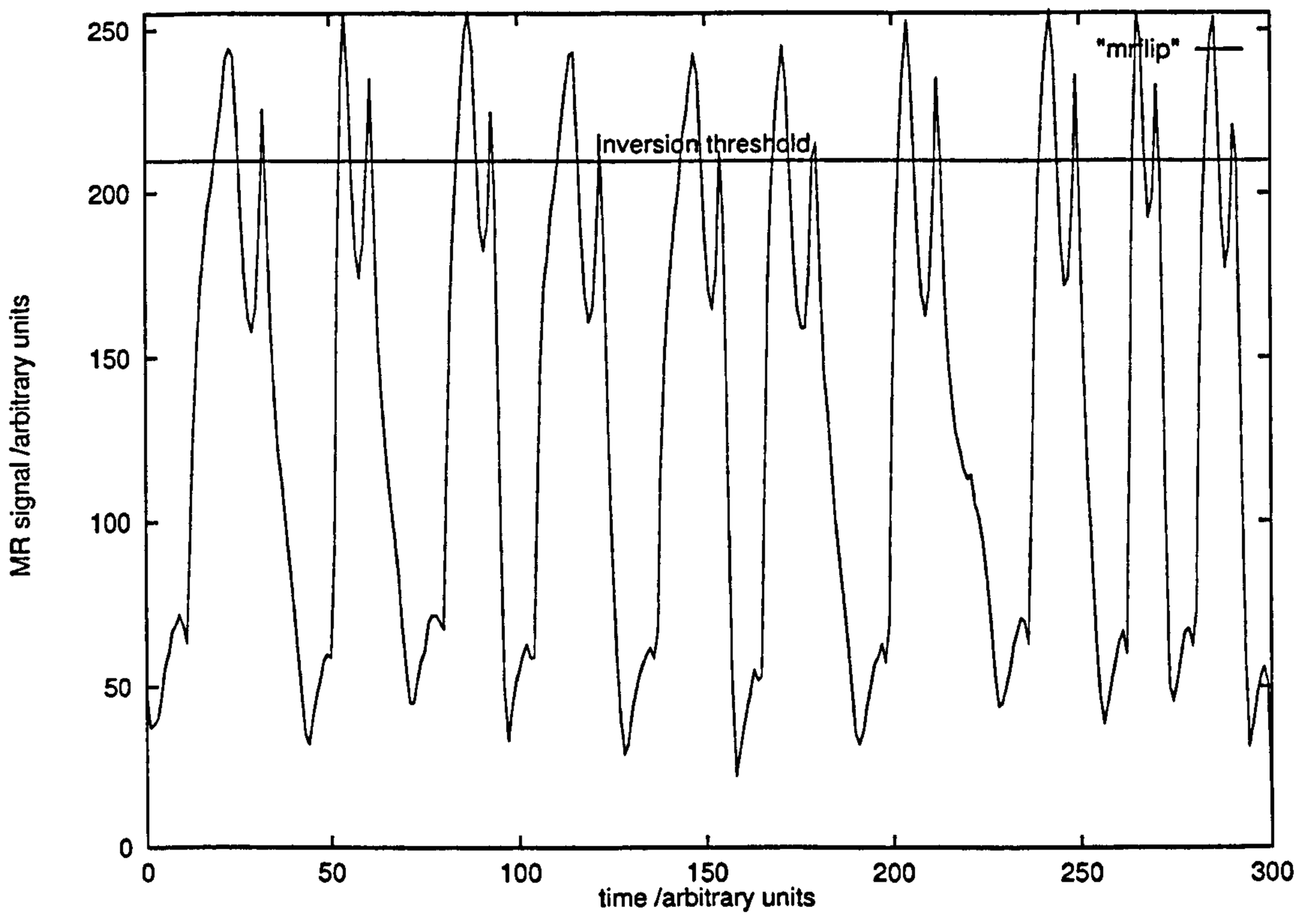


Figure 1.9: MR read-back signal showing inversion characteristic (flip-back).

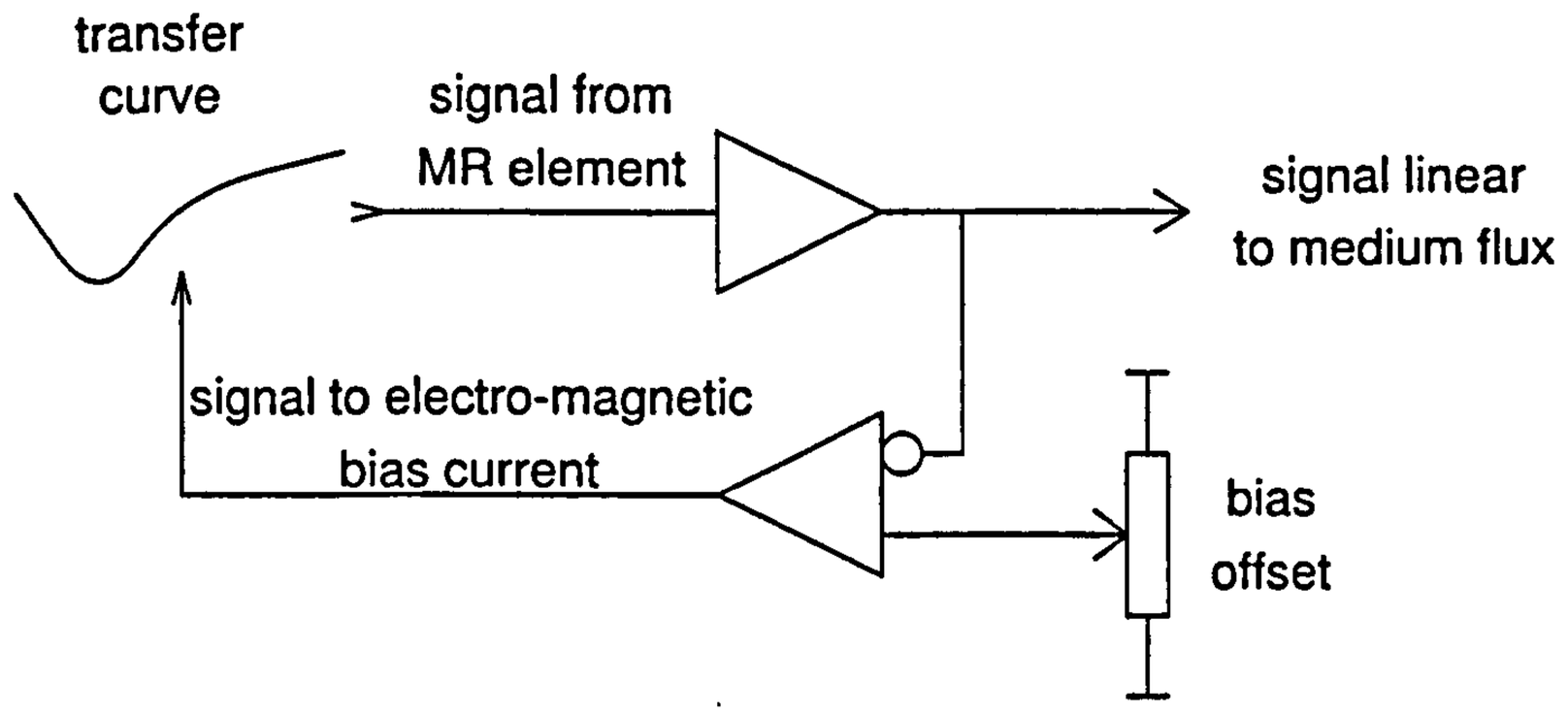


Figure 1.10: Schematic circuit to make a single MR head linear.

One simple method to prevent inversion is increasing the recording density and slightly adjusting the head-medium spacing to attenuate the medium field seen by the head. This is not ideal for a retail system for two reasons. Firstly information capacity is reduced by removing the head from the medium. Secondly the card may already be slightly removed from the medium because of a poor swipe, and hence the read-back amplitude would be low. Ultimately these severe non-linearities were somewhat compensated computationally during the detection phase in this study.

Linearizing the MR element

DCC: Digital Compact Cassette

Figure 1.10 shows a circuit schematic that will make an MR head linear. It is used, for instance, in the DCC analogue channels [78, 79]. This system inherently avoids the non-linear properties of an MR element by making the amount of magnetic flux through the element 'constant' by modulating the bias field. This method is not feasible in a closely-packed multi-track system that has only one bias coil common to all heads.

Detection of flip-back

Tightly packed MR elements in a head assembly tend to use a common bias line for cost and complexity reasons, so in this case individual bias control is not possible. Fortunately, since inversion always occurs at the same signal level, it is a relatively easy task to detect where inversion may be taking place. It is very difficult to reliably correct this problem though. (If the signal drops down quickly after hitting the trigger level, then inversion did not occur. If the signal quickly climbs back to the trigger level, then it is likely that inversion did take place. However, the recorded data could also cause this effect, so post-read correction is tricky.)

1.1.10 Summary of magnetic recording

The process of magnetic recording can be considered as being the record of the direction of the recording magnetic field when it fell below a threshold. The medium itself comprises many grains that each individually record the pattern. The grains maintain the direction of the recording field when it dropped below critical strength in their own ferromagnetic field (at least in isotropic media).

A magnetic read head can detect a component of the magnetic field that extends out of the medium that is in turn formed from the superposition of the grain magnetic dipole moment-like fields.

There are two common forms of magnetic read head: the inductive coil, which reads linearly in rate of change of magnetic field, and the magneto-resistive (MR) read head, which reads a non-linear function of the magnetic field.

1.2 The credit card standard

Plastic swipe cards have been in circulation for over 20 years. They are so common that a de facto format became an international standard [13]. (Further relevant standards documents are listed in table 1.1.) The principal purpose of

Identification Cards—Financial transaction cards	ANSI/ISO 7813–1987
Identification Cards—Recording technique—part 1: Embossing	ANSI/ISO 7811–1–1995
Identification Cards—Recording technique—part 2: Magnetic stripe	ANSI/ISO 7811–2–1995
Identification Cards—Recording technique—part 3: Location of embossed characters on ID-1 cards	ANSI/ISO 7811–3–1995
Identification Cards—Recording technique—part 4: Location of read-only magnetic tracks—tracks 1 and 2	ANSI/ISO 7811–4–1995
Identification Cards—Recording technique—part 5: Location of read-write magnetic track—track 3	ANSI/ISO 7811–5–1995
Identification Cards—Identification of issuers—part 1: Numbering system	ANSI/ISO 7812–1–1993
Identification Cards—Identification of issuers—part 2: Application and registration procedures	ANSI/ISO 7812–2–1993

Table 1.1: Standards relating to magnetic transaction cards.

the magnetic stripe is to provide an account number to the retailer without typing. The retailer is trusted to pass this and billing information on to the credit card company, which then reimburses the retailer and charges the customer.

The account number is normally 20 digits or so (about 56 bits) along 3 inches of magnetic medium. Additional information and abundant, though simple, error-checking bits are also stored in the stripe.

The standard uses non-SI units, and measurements in this section are reproduced using the standard units.

1.2.1 The magnetic medium

The ISO standards tightly define card dimensions, warpage, burring, thickness, finish and smoothness of the card and magnetic medium.

However, the medium's magnetic properties are only defined in terms of a 'reference tape signal level' stored at 200 flux reversals per inch (frpi) and a set

frpi = flux reversals per inch

of guidelines [3]. The read-back signal level must be between 80 and 130% of this reference level. At 500 frpi the signal must be better than 70% of that at 200 frpi (using a read head with a gap width of 0.5 milli-inches (min) or 13 µm).

min = milli-inch

As such, the coercivity of cards in circulation varies from 300 ('lo-co') to 3000 ('hi-co') oersteds. It also allows for manufacturers to cover the medium with a material layer providing protection against wear and scratching up to around 2 min thick. This poses significant problems for high density storage, as the variation in head-medium distance causes large amplitude swings at thousands of frpi.

1.2.2 Stripe layout

There are three tracks (figure 1.11) defined for a standard banker's credit card, each 110 min wide, separated by guard bands of 20 min. The central track (track 2) is the most important: the linear data density there is 75 bits per inch (about 0.6 kbit/in^2). The two outer tracks (1 and 3) store additional (optional) data at 210 bits per inch (about 1.6 kbit/in^2). Track 3 was originally designated for erasable data, but it is not in widespread use because of historical security problems.

Information is recorded longitudinally with a wide-gap head and no azimuth offset angle. It is possible, though rare, to rewrite cards once they are in use.

1.2.3 Coding scheme and track layout

FM: frequency modulation

All tracks use FM (frequency modulation) code (also called two-frequency coherent phase encoding) to represent bits (figure 1.12). With this code there is a magnetic transition at each bit cell boundary which can be used to recover a data clock. A `1` bit is denoted by a transition at the bit-cell centre. At the start of each track, there is a preamble of around twenty `0` bits, to allow the clock to synchronize and any automatic gain control to settle.

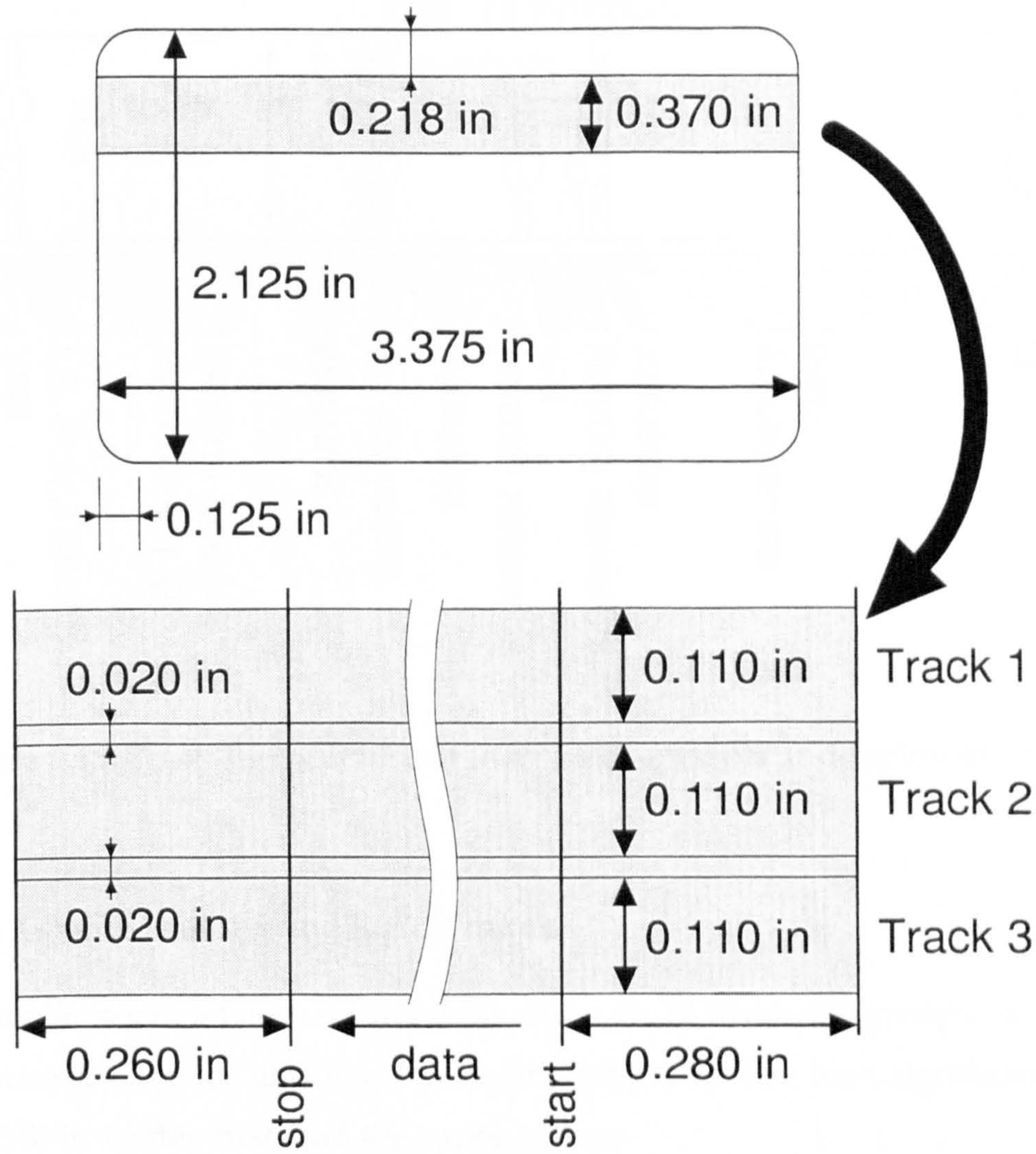


Figure 1.11: Positional layout of tracks on ISO standard magnetic stripe card.

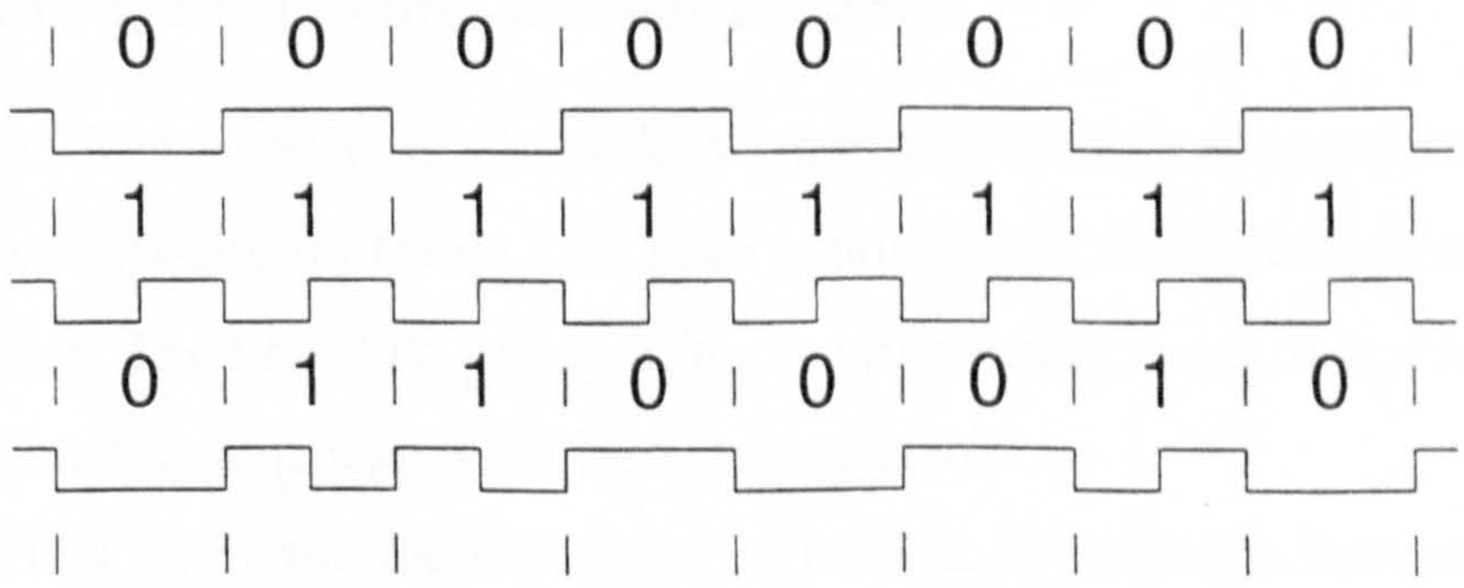


Figure 1.12: FM coding signals with example data.

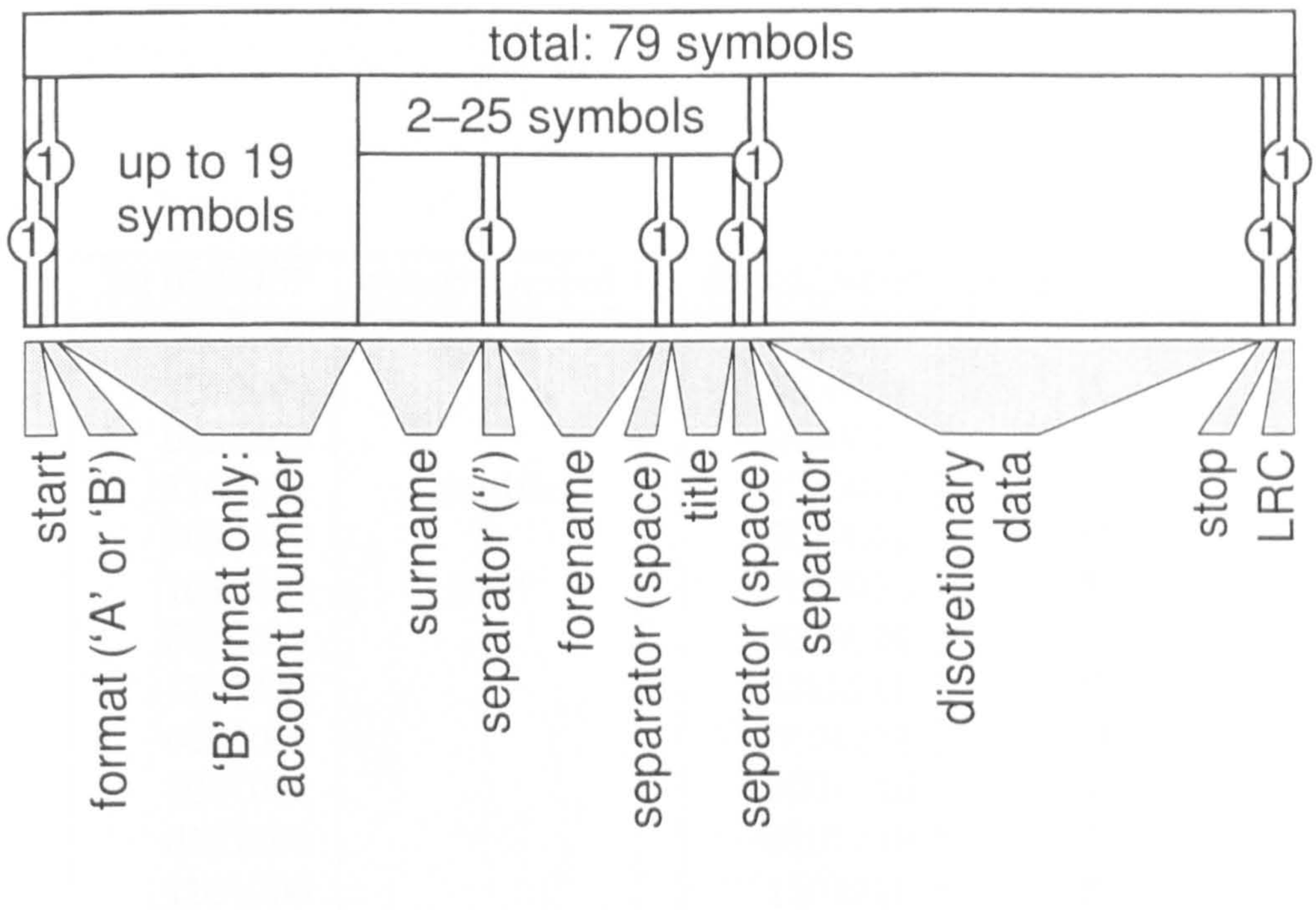


Figure 1.13: Track 1 logical format in terms of symbols and typical example.

Track 1 symbol coding and logical format

This track stores letters and numbers in up to 79 seven-bit groups: a group comprises 6 data bits and 1 odd parity bit (table 1.2). The least significant data bit (LSB) is written first, and the parity bit last.

LSB: least significant bit

Figure 1.13 shows the logical format of track 1: it principally stores the account number and the name of the card holder.

Track 2 and 3 symbol coding and logical format

Tracks 2 and 3 store only numbers. Bits are grouped in fives comprising 4 data bits and 1 odd-parity bit (table 1.3). Each group is laid down LSB first, with the parity bit last. The first data group (after the preamble) must be a start symbol, and the penultimate group must be a stop symbol.

Figure 1.14 show the logical format of track 2. It provides the card account number, expiration date and a longitudinal redundancy check (LRC: even cross parity—the exclusive-or of all the data symbols).

LRC: longitudinal redundancy check

Bit 012345P	group symbol
0000001	space
1000000	
0100000	
1100001	graphic
0010000	$
1010001	start
0110001	
1110000	
0001000	(
1001001	)
0101001	
1101000	
0011001	
1011000	–
0111000	.
1111001	/
0000100	0
1000101	1
0100101	2
1100100	3
0010101	4
1010100	5
0110100	6
1110101	7
0001101	8
1001100	9
0101100	
1101101	
0011100	
1011101	=
0111101	
1111100	stop

Bit 012345P	group symbol
0000010	
1000011	A
0100011	B
1100010	C
0010011	D
1010010	E
0110010	F
1110011	G
0001011	H
1001010	I
0101010	J
1101011	K
0011010	L
1011011	M
0111011	N
1111010	O
0000111	P
1000110	Q
0100110	R
1100111	S
0010110	T
1010111	U
0110111	V
1110110	W
0001110	X
1001111	Y
0101111	Z
1101110	reserved
0011111	reserved
1011110	reserved
0111110	separator
1111111	

Table 1.2: Track 1 code symbols.

2^0	2^1	2^2	2^3	parity	group symbol
0	0	0	0	1	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	1	3
0	0	1	0	0	4
1	0	1	0	1	5
0	1	1	0	1	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	1	9
0	1	0	1	1	
1	1	0	1	0	start
0	0	1	1	1	
1	0	1	1	0	separator
0	1	1	1	0	
1	1	1	1	1	stop

Table 1.3: Tracks 2 and 3 code symbols.

Track 3 is not rigidly defined: it can store up to 107 groups. Its logical format is controlled by another standard [14].

1.2.4 Types of reader

There are four main types of card reader: the hand operated devices (slot-swipe and wand) and the motorized devices (moving card and moving head). Each of these give roughly the same reliability when used properly. Hand swipes are the cheapest and most common form of reader, but the motorized form is used when tampering or security is a problem. Motorized forms can give rise to small amounts of jitter, depending on the mechanism (belt driven readers especially can suffer if they are not serviced). Hand swiping gives rise to large speed, azimuth and head-medium separation variations when reading; making it relatively difficult to write to a card in a hand operated mechanism. Where hand swipe writers do exist, they tend to be expensive and often use a separate track for clock information.

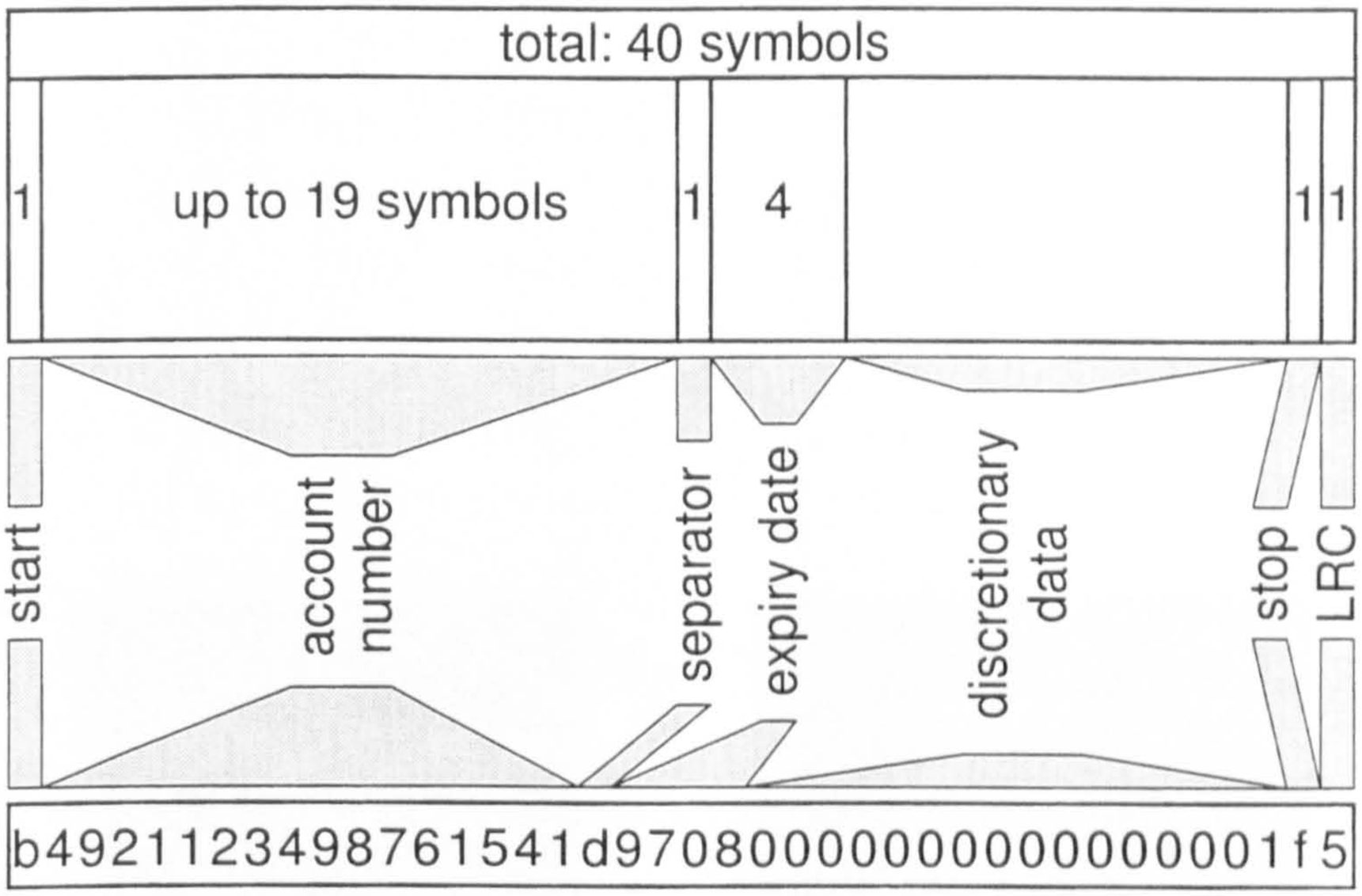


Figure 1.14: Track 2 logical format in terms of symbols and typical example.

Recently a class of insert-then-remove hand operated reader has become popular. These can also write to cards if a suitable clock can be recovered from a reference track. With untrained users, this form gives a more reliable read than the slot-swipe form because the card is tightly constrained as it enters and leaves the slot. The card is read during the withdrawal phase, when its acceleration is much smoother than on entry.

Four commercial credit card readers (1 insert-then-remove and 3 swipe) are shown in figure 1.15. The units have a variety of electrical interfaces, from a simple amplifier to complete microprocessor decoders. The units also use elaborate head mounts to improve read reliability. The head suspension applies pressure to the head to keep it in contact; and the head can always rotate a small amount to preserve contact across the card in case of warping or undulation.

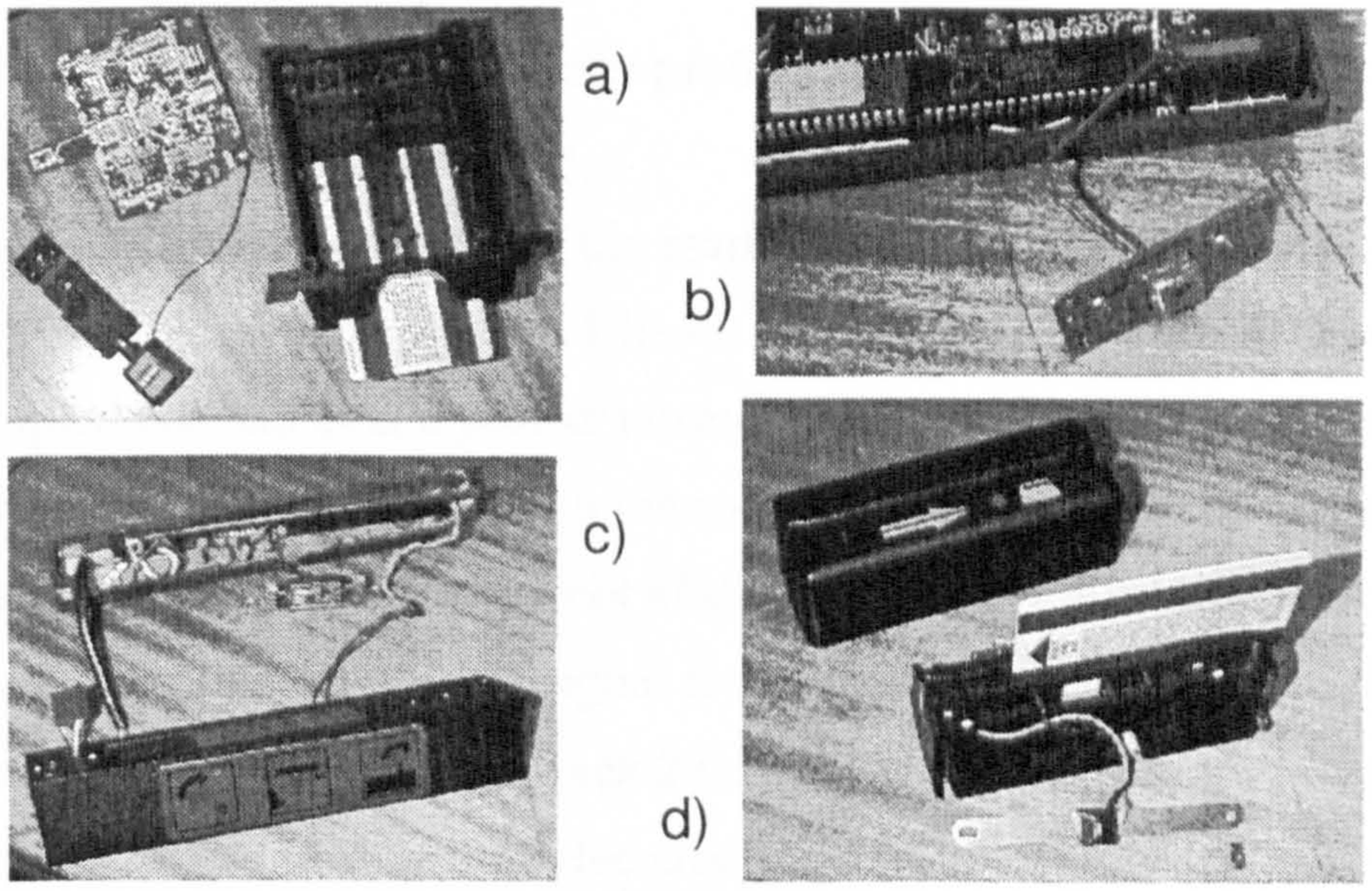


Figure 1.15: Commercial card readers and their head mountings. a) dual-stripe insert-then-remove reader and writer: head can rotate on sprung strip. With on-board microprocessor. b) ICL reader: leaf spring mounted on pivots. With on-board microprocessor. c) telephone swipe: leaf spring mounted head in rotating assembly. Output is amplified (digital) voltage from head. d) MR swipe: head mounted on single long leaf spring. Clock and data digital outputs.

1.2.5 Reading properties

The following results are not strictly part of the credit card standard, but are representations of the environment that the credit card reader and decoder must withstand. The two paramount factors are the card's velocity profile through a reader (which affects the signal amplitude for an inductive read head) and the quality of the read-back signal itself.

A hand swipe reader must cope with a range of card velocity, and changing card velocity during a swipe. The variation may be a few hundred percent. Figure 1.16 shows some card velocity profiles measured with the swipe reader described in section 2.3.1 (page 46).

Figure 1.17 shows a portion of the read-back signal (a value that is proportional to the voltage from the read head) from a real credit card. The credit card sampled has had over a year of normal usage and has been in the author's pocket for most of its life. Although the surface of the magnetic stripe is visibly scratched and blemished this has not affected the magnetic information to any significant degree. This type of signal is indicative of what must be decoded from a swipe card mechanism. Track 2 (150 frpi) should always be decodable, but track 1 (420 frpi) is often only decoded by mechanical readers.

frpi: flux reversals per inch

Both signals in this instance are clear and strong enough to be decoded with a simple threshold peak detector (section 5.4.1, page 122). The FM code makes the timing easy: the track 1 inter-peak interval from a threshold peak detector is shown in figure 1.18. The clock sentinels (all zero bits before the start and after the end of the data) are also easy to pick out in this chart.

1.3 Competition to the magnetic stripe card

One does not have to look hard to find storage technologies that are directly comparable to the plastic swipe card. The two most immediately obvious are the printed bar code and the Smart card. Punched cards are now almost completely removed from public circulation because of poor wear properties. Phase

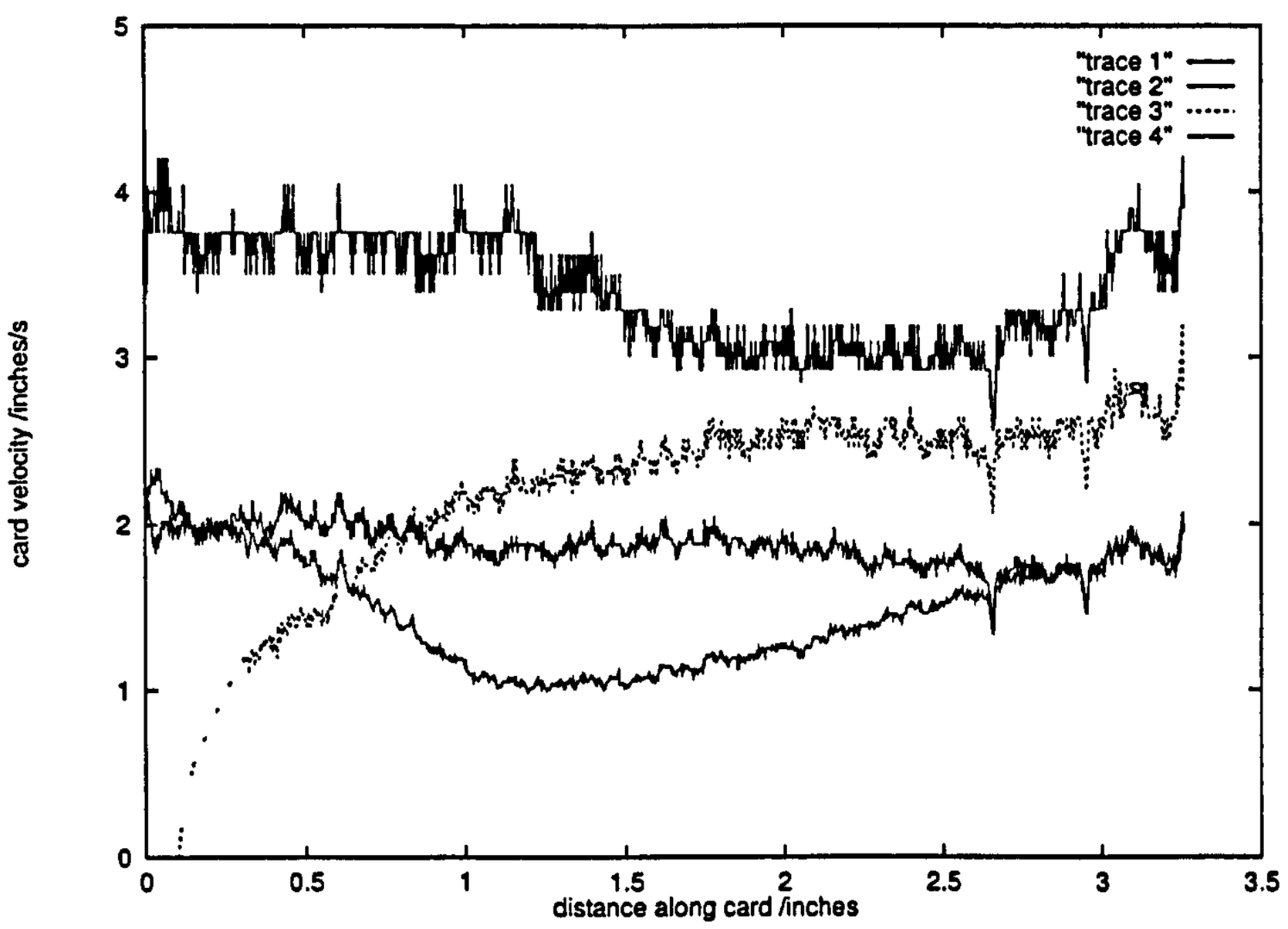


Figure 1.16: Example velocity profiles of cards being used in a hand swipe reader, calculated from a recorded clock track. One profile (trace 3) was started from zero velocity a small distance into the card. There are index marks around 2.7 and 3.0 inches. The speed dependent noise in the velocities is caused by signal sampling (the period detection algorithm only places cycles to the nearest quarter sample: the sample rate was constant for all four swipes). There is other jitter noise caused by the writing process (mechanical, temporal, field induced and magnetic, etc.), and the magnetic particle size.

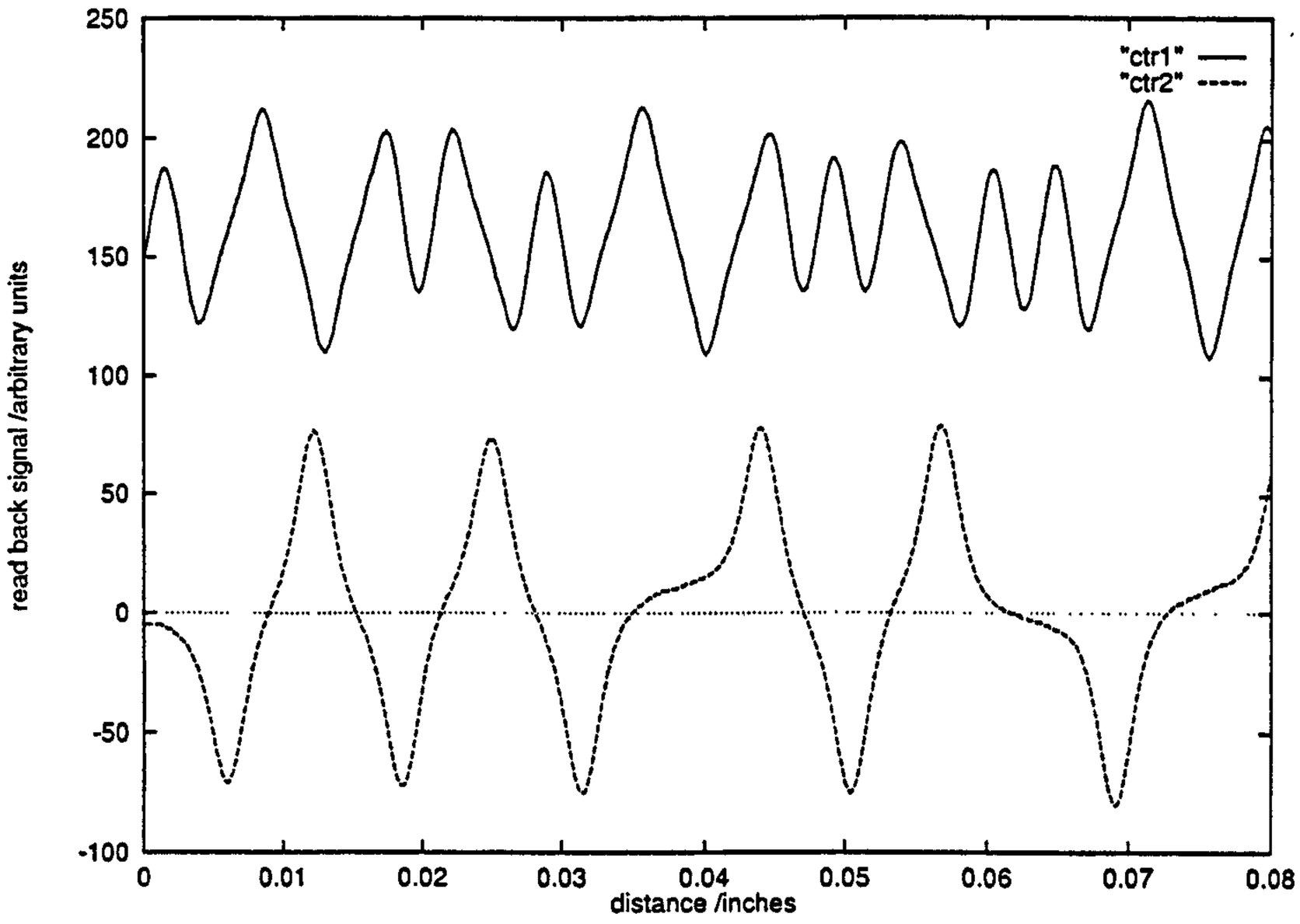


Figure 1.17: Example raw read-back signals (amplified voltage) from tracks 1 (top, 420 frpi) and 2 (bottom, 150 frpi) from a one-year-old bank-issued credit card. Read with an inductive head.

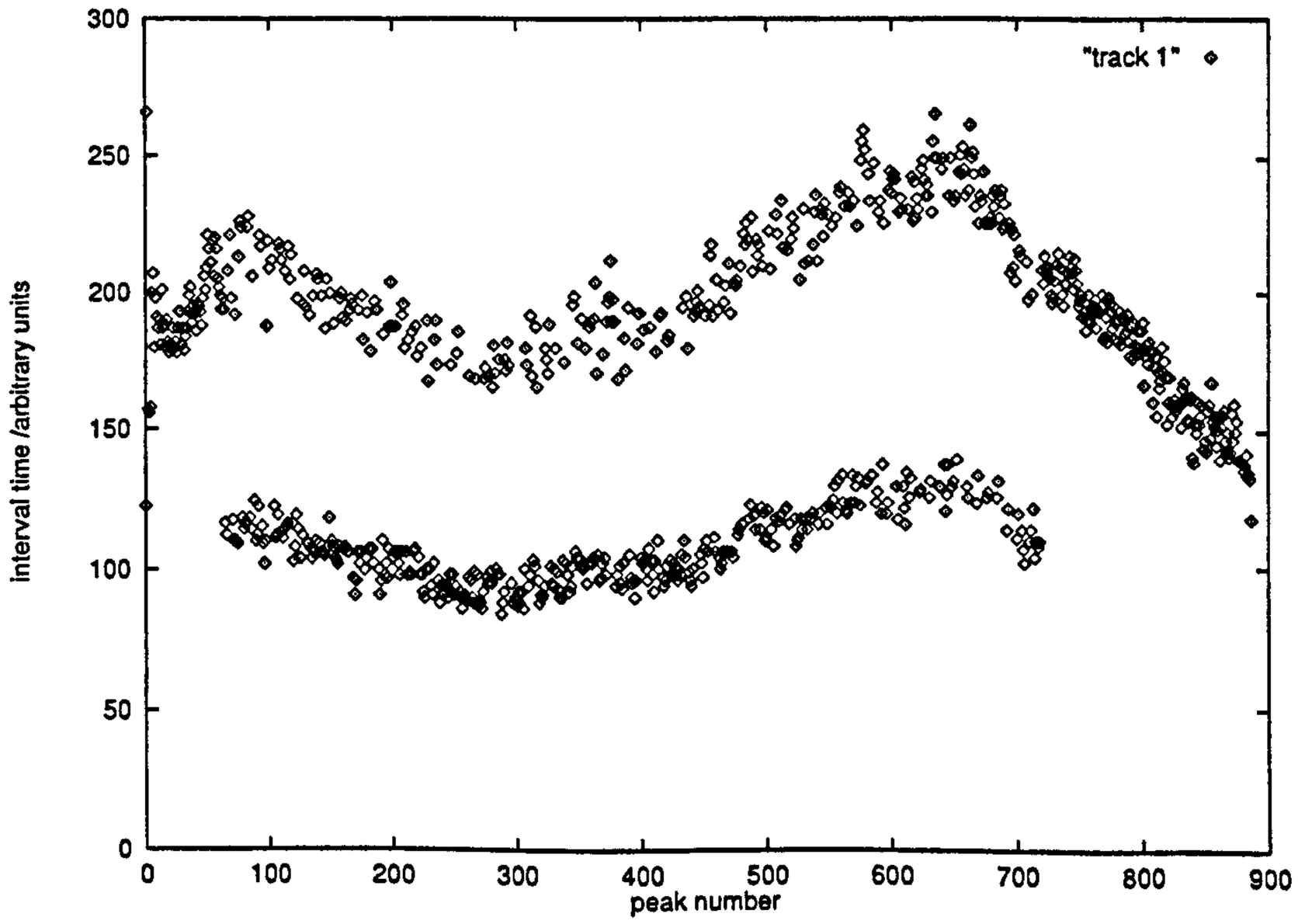


Figure 1.18: Interval between each peak from track 1 of a standard credit card.

	mag. stripe card	Smart card	printed bar code
Cost /pence	~10	~100	~0
Cost of reader /£	~20	~20	~20–200
Capacity /bytes	137	64–16 k	4
Copyability	easy	difficult	trivial
Tamperability	moderate	difficult	moderate
Writable	not usual	yes	no
Durability	2–3 years	2–3 years	indefinite
Uses	credit card i.d. card	credit card electronic cash	product i.d. i.d. token
Special features	watermark embossing	encryption keypad microprocessor	2D-codes remote reading easily created

Table 1.4: Features of the magnetic strip card, the Smart card and the bar code (approximate).

change optical cards are specialized, but they can store from a few bytes up to a few megabytes, depending on cost. Although each system can have special versions, the general features of the three major competing portable data storage devices are shown in table 1.4. The disposable paper magnetic stripe tickets have not been included in the table, but are an even cheaper version of the plastic stripe card that can only be used a few times. Radio tags may also become more widespread in the future. Another possibility is that these token items will disappear, to be replaced with a central database and biometric transducers. Whatever happens though, the magnetic stripe plastic card will undoubtedly hold some niche for the foreseeable future.

1.4 Uses of a high density magnetic stripe card

There are vulnerabilities in the security model of the standard credit card. The card is moderately easy to copy (or even to design a new one). Telephone sales are difficult to authenticate. Automated card readers cannot normally tell one magnetic stripe from another—in other words the contents of the stripe can be

transferred between different cards. And if a card is stolen, then a thief can use the card, copying any signature that may be required, until a retailer identifies the card as invalid; a procedure that requires querying a central database.

Although a high density card cannot solve all these problems, it can at least increase the difficulty of copying the magnetic stripe (as there will be more bits to copy) and of inappropriately using the card (because there can be more identifying data in the magnetic stripe). Watermarking [56] (*printing* different orientations of high coercivity medium within a low coercivity stripe) and fingerprinting [45] (identifying a card from the precise pattern of jitter in the magnetic signal) can stop this transfer of the magnetic information if appropriate readers are used.

A high density card can also make casual fraud much harder. Consider the difficulty of forging a signature if that signature is not printed on the card but stored within the magnetic medium itself. Combine this feature with a photograph of the authorized user, also within the magnetic stripe, and opportunistic theft starts to become uneconomical.

A digitized signature can be stored in about 1 kByte of data and a passport sized colour photograph in about 2 kBytes. Fingerprint information (from people's fingers) can be stored in a few hundred bytes and the entire fingerprint in about 4 kBytes. A secure public encryption key can be stored in fewer than 200 Bytes. A standard credit card stores 137 Bytes, of which only 20 are read in a typical transaction.

1.4.1 Electronic cash

There have recently been trials to test the feasibility of electronic cash—a replacement for notes and coins [68]. This utilized a Smart card system because of the security requirement (each card contains a microprocessor, and transactions are completed in a secure and encrypted fashion to another similar microprocessor) and so that each card can store its current value. Although it has been shown that there are weaknesses in the technology used to implement the

encryption algorithms and storage unit [4, 59], electronic cash is a practicable and real development, after all, even real cash has weaknesses. However there is nothing fundamental to prevent electronic cash becoming 'economically' secure.

The relevance of electronic cash to the credit card is that there are versions of the cash card with credit facilities as well. It is reasonably likely that the combined card will make the magnetic stripe card obsolete for financial transactions in the mid to long term future.

1.4.2 Security

It is useful to think of a payment medium as 'secure' when the cost involved in defrauding it is greater than the reward obtainable from the fraud. So with electronic cash, which is anonymous, almost untraceable and capable of a large turnover, a great deal of effort has gone into making the system secure. The credit card on the other hand is attributable, audited and limited in transaction size. This has made the credit card per se reasonably immune to large scale fraud even with almost no security features. In Europe in 1995 the fraud level was reported at 0.04% of turnover (US$120 M) [38]. Because the credit card is based on *credit* given by the major card companies and retailers, when fraud is discovered the losses can be contained.

Small scale fraud on the other hand is much more common. The credit system itself relies on a number, a name, an expiry date, an optional signature and an occasional photo-i.d. all printed for inspection on the card. The card itself however need not even be present to perform a transaction. Both the signature and photograph, if displayed, are vulnerable to simple duplication during a transaction. If such identity marks could be hidden from view and encoded in the card within, say, the magnetic stripe then this simple copying will be difficult. Tampering is another avenue of attack. The information within the stripe should not be alterable and still remain valid—a (now) mature technology of public key cryptography [25] is easily capable of this. The magnetic stripe is,

however, still vulnerable to copying: watermarking [56] and fingerprinting [45] can prevent successful card duplication. However, this still leaves organized fraud committed by retailers. A retailer could store the information about a card when it is used, and replay it later thus making additional transactions. This type of fraud cannot be prevented without a central on-line database of transactions. The central server model does not demand a high density credit card as biometric information can be transferred to the retailer's verification equipment over the communications link and the card itself only serves as an i.d. number. When retailer fraud is detected, legal proceedings and financial penalties can be used against the rogue company.

It is thus sensible to presume that a high density credit card would place itself in the retail network where an on-line connection is either not desired or is uneconomic, or where the retailer can be trusted (e.g., large retail organizations).

1.4.3 How much capacity is required?

A passport-sized colour photograph can be stored in about 2 kBytes and a signature in 1 kByte. The desired total card capacity is therefore on the order of 3 kBytes. At a linear density of 200 bpi this requires approximately 40 tracks of data (16 mm width at 0.4 mm per track). To achieve the same capacity in 4 tracks requires a linear density of 2000 bpi. The credit card in its current form is not capable of replaying this density. At best, 1000 bpi over 8 tracks will be possible, but this density will later be shown to be impractical at present due to medium induced jitter considerations.

1.5 Errors: detection, correction and concealment

Virtually all modern-day high density digital communications systems (telephone modems, radio data links, satellite broadcasting, storage on tape and disc whether read/write or read only) use some form of error correction (EC).

The overhead introduced by adding these EC schemes is overcompensated by allowing the medium to work at higher densities than would be comfortable at lower, 'error free', densities. In two-way communications (e.g., two connected modems) error detection—using additional check bits in the data stream—can perform EC by asking for the corrupted data packets to be retransmitted. In one-way communications (broadcasting and all storage media) forward error correction (FEC) must be used. An FEC code (also commonly called an error correcting code, ECC) adds check bits to the data stream that can detect an error and also correct certain classes of bit errors. The proportion of check bits added and their error correcting ability is dependent on the ECC method used, but a typical system might reduce an error rate of 10^{-5} to around 10^{-12} with a 15% overhead [5, 20].

When the ECC fails, the last resort is error concealment. If the bits on the medium have been interleaved or scrambled then an error that makes a data block or symbol unreadable will (hopefully) be immediately adjacent to correct data. Then the incorrect data can be interpolated from the correct, masking the fact that no usable information exists at that location. In sampled audio recording a single missing sample can be interpolated by pretending that the Nyquist frequency has halved and interpolating the missing sample. The consequences are that the error is, audibly, undetectable. Some very simple error concealment can be applied to transmitted photographs that suffer large block errors: scrambling the order of the 8×8 pixel blocks of the JPEG standard [97] and interpolating the corrupted blocks [15] can produce results not easily discernible from the original. Nevertheless burst errors or statistically improbable error patterns can overcome the interpolation system from time to time. In this case, for storage media, rereading may be in order. For broadcast media, muting or freezing may be the only option.

In this investigation error detection, correction and concealment have not been studied in depth. Although an ECC is necessary for a practical high density credit card, and indeed even the standard credit card has 20% overhead for error detection, the scope of the subject is too broad to cover this important

ECC: error correcting code

communication facet satisfactorily.

1.6 Ethos and contribution to knowledge

The techniques necessary for performing high density storage on a magnetic stripe card have been investigated and developed. The term 'high density' is taken to mean recording where linear bit densities are high enough to cause inter-symbol interference. That is to say, 'high density' implies that the data density limits of the medium are being striven for.

The aim was to record on and read back from a credit card using high density recording techniques that has a 'reasonably low' raw bit error rate on read-back. The cost of a reader mechanism must be low. The cost of a card writing system was irrelevant (the credit card is a write once medium with few writers and many readers). In addition, the card should be capable of being read in a hand swipe mechanism.

It is interesting to note that there are *no* openly published methods that can achieve this aim. There are however many published results in related fields, all specialized to particular systems and media that are very dissimilar to the credit card. There has been work done by other parties on increasing the data density of credit cards, but these have not been taken up in any great way, and do not represent a high density solution (and the results have not been generally published). Again, because the credit card has become a de facto standard, the cost of physical manufacture of the cards has reduced to just cover the tolerances the card standard demands. The driving force in this business has been to reduce the cost *per card* to store a fixed number of bits, as opposed to the storage industry in general that is driven to reduce the cost *per bit*, which is achieved through tight tolerances and high densities.

The contributions to knowledge described in this thesis are presented in their contexts. The principal new idea is the operation at variable speed of a high density Bayesian signal detector (sections 4.2 and 5.5.3). Additionally a new measure of code error resilience has been devised (section 3.2), which also

required work on the strategy for recovering from code timing errors. A new code was devised to help solve the problem of synchronizing after timing errors in a variable speed environment (section 3.3). A further contribution is the extension of the Bayesian detector to partially cover non-linear distortion channels through the use of probability modulation (sections 4.2.3 and 5.5.4).

1.7 Organization

This thesis is split into several major sections, with results being presented when appropriate. Section 1.1 provided the physical background and justification for high density magnetic recording. The credit card standard was given for reference in section 1.2. This examined the current credit card technology. The apparatus used for experimentation is described in chapter 2. Density increases have been achieved by raising the track density as well as the linear density. The track density is discussed in section 2.4 while linear density, coding and detection techniques, are discussed and derived in chapters 3 and 5 respectively. In addition the theoretical limits to recording density are discussed in section 5.1. All detection systems need a timing clock recovery system, which is treated in chapter 4. Finally a summary of the high density credit card and conclusions are presented in chapter 6.

Relevant published papers by the author are reproduced in appendix A. The remaining appendices cover subjects that can be treated on their own; separate from the dialogue of the main text. Appendix B derives the spectral properties of codes and code efficiency. It provides a justification for the use of the spectral shaping of raw codes (i.e., without partial response channels) and efficiency measurements. Appendix C describes and details the electronic equipment built for the project while the most interesting software developed for the project is listed in appendix D. Following these, references and an index are provided for convenience.

Chapter 2

Experimental apparatus

The physical apparatus comprises two separate and independent units: the card reader and the card writer.

It was anticipated that in a commercial situation there would be few card writers (belonging to the card producer) and many card readers (belonging to retailers). To this end the card reader has been designed to be simple and robust with minimum required maintenance. No restrictions were applied to the card writer.

As a starting point, the reader and writer must both be able to manipulate the standard credit card format.

2.1 Constraints from the standard card format

A standard credit card uses 3 parallel tracks of longitudinally recorded data (section 1.2 gives details). The entire contents of the stripe can be read in a single swipe with a suitable reader.

swipe: pushing a card through a reader mechanism by hand

The linear data density is different on the three tracks, so it is better that the read and write heads are not tuned to a specific bit density. For instance, a narrow-gap write head is unsuitable for writing low density information because the write field will not penetrate the medium deeply enough. On the other hand a wide-gap write head will have larger lateral fringing fields and a less well defined trailing magnetic field. Read heads are not so critical, as gen-

erally a head designed for high densities can also read low density information.

The speed of the card through a hand reader can vary, so the read system must not be compromised by changes in card velocity. These changes can be quite substantial (more than 100% variation in a swipe, more than 200% in an insert-then-remove reader) but they are reasonably gradual on the scale of the bit length.

Finally, there is a protective layer over the magnetic medium in the stripe. Although this does not affect the signal from the read head, it does place a minimum gap width constraint on any write head. If the critical magnetic field protruding from the write head is shorter than the 50 µm needed to penetrate the recording medium then the recorded signal level on the stripe will vary enormously (see, for example, figure 2.6, recorded with a head with a ~3 µm gap). As a yardstick, 50 µm corresponds to a linear density of about 500 frpi before inter-symbol interference (ISI) becomes very prominent.

frpi: flux reversals per inch
ISI: inter-symbol interference

2.2 Write mechanism

The write mechanism is split into three main parts: magnetics, electronics and mechanics. The magnetic features are influenced by the write head. The electronics control the head signal and the mechanics control where the head writes on the card.

The overall design is a hybrid stationary single head, moving card system. The head can move laterally, but is longitudinally fixed. Information is written digitally, in either a pulsed or continuous fashion.

2.2.1 Moving card writer: mechanics

The write system was built around a surplus dot matrix printer (figure 2.1). The magnetic head is fixed over the carriage, which carries a stripe card in place of a print head. The printer's on-board processor controls the card position with a stepper motor, with simple commands being sent from the controlling

computer via a modified parallel port [27]. The speed of movement is selectable at 5 or 10 in/s. Physically, the card carrier is connected to the stepper motor via a toothed belt and is controlled to a resolution of 1/240 in (about 0.1 mm).

in/s: inches per second

The printer's paper feed motor controls the head's lateral position through a geared screw assembly. This gives about 4–5 mm of movement controllable to a resolution of 0.71 µm (28 µin) with unmeasured repeatability. This arrangement is shown schematically in figure 2.2 and as a photograph in figure 2.3. It allows several tracks on the card to be written during multiple passes under the head: each time the head position is adjusted one track width. The head rotates slightly as it is moved, introducing an insignificant azimuth misalignment of about 3°. (With a 500 µm × 3 µm read head, 3° of azimuth gives a maximum effective head width of about 30 µm, equivalent to a spectral null point of around 850 frpi—significantly greater than the limit imposed by the head-medium separation.)

Small infra-red detectors attached to the head are used to tell the controlling computer when the card is in position. It is possible to fit two heads over the carriage, and the hardware allows up to three position detectors to be used. This flexibility allows a read or erase* head to be positioned next to the write head, which is useful for unattended and repetitive operation.

All heads are spring mounted to allow continual close contact with the card, and it also allows the head to rotate slightly in order to stay flat against the card as it moves past. Experiments have shown that the recorded signal level is extremely sensitive to variations in the card-head angle, mostly due to the profile of the head causing the active portion to separate from the surface giving increased spacing loss.

2.2.2 Write electronics: head amplifiers and FIFO

The full circuit diagrams for the 8-track parallel write system are given in appendix C. Essentially the write electronics comprise an 8-bit digital data stream,

*A DC erase was used, by mounting a permanent magnet above the carriage.

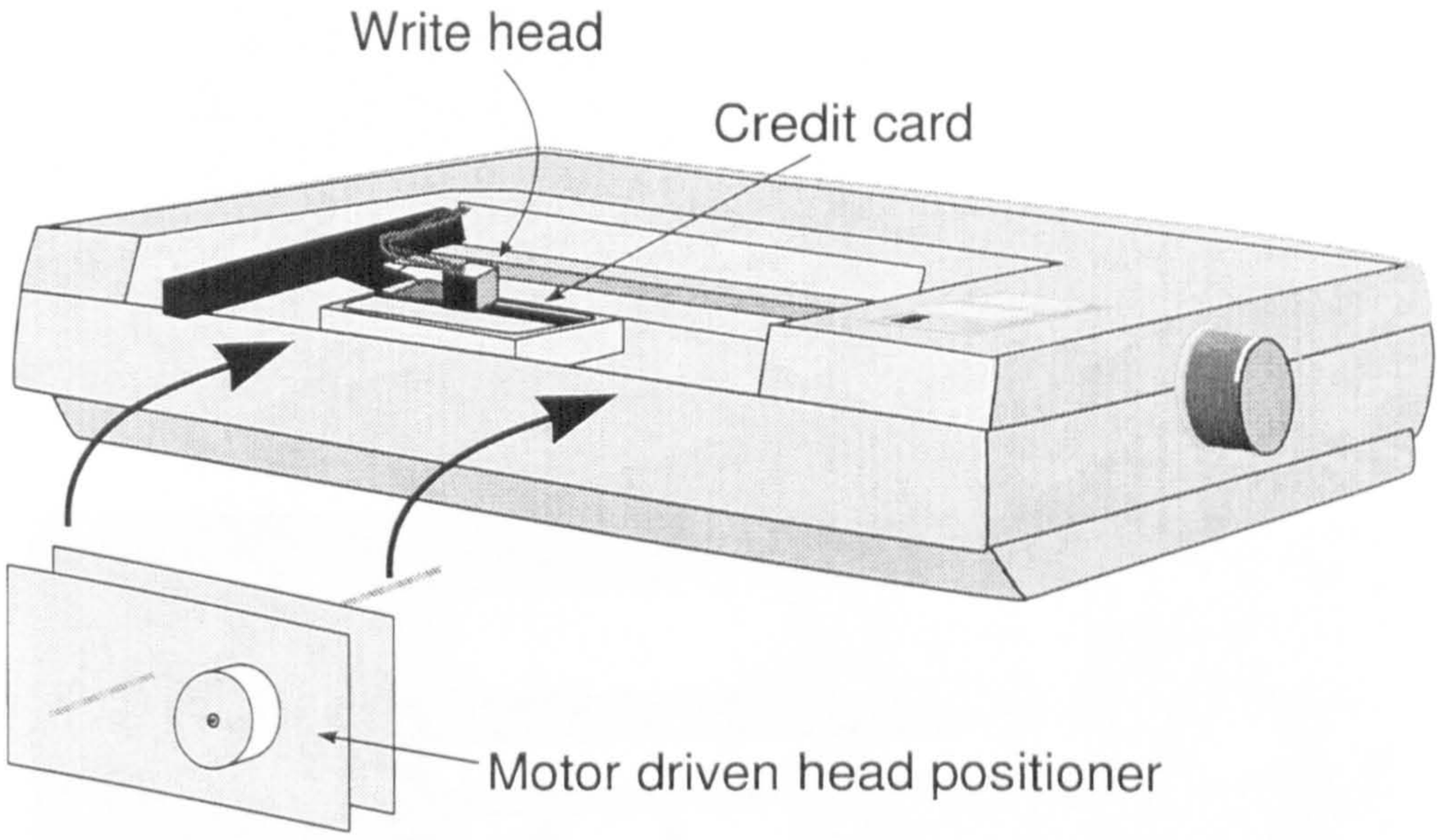


Figure 2.1: Schematic view of write system mechanism.

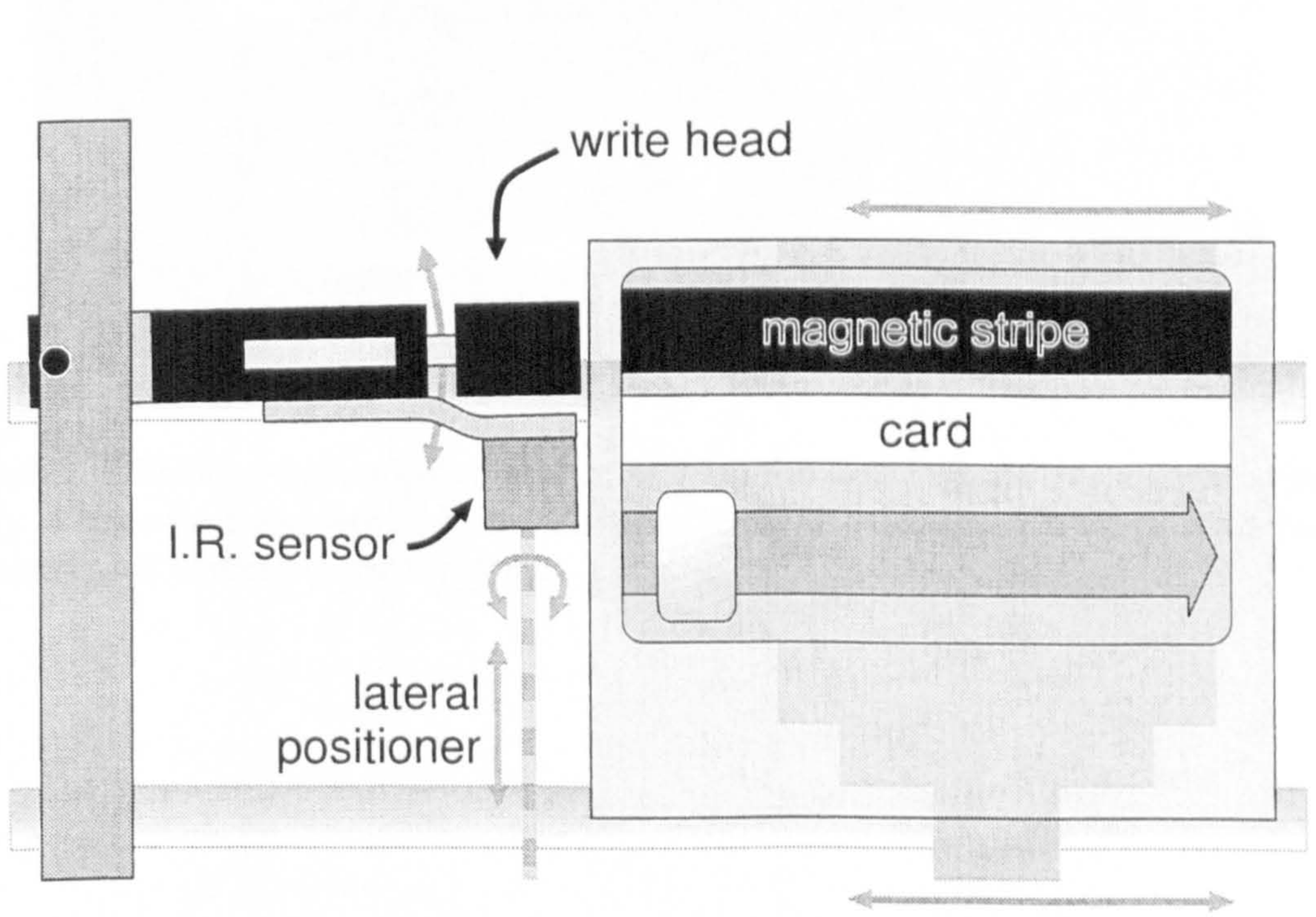


Figure 2.2: Overhead schematic view of write system showing movement.

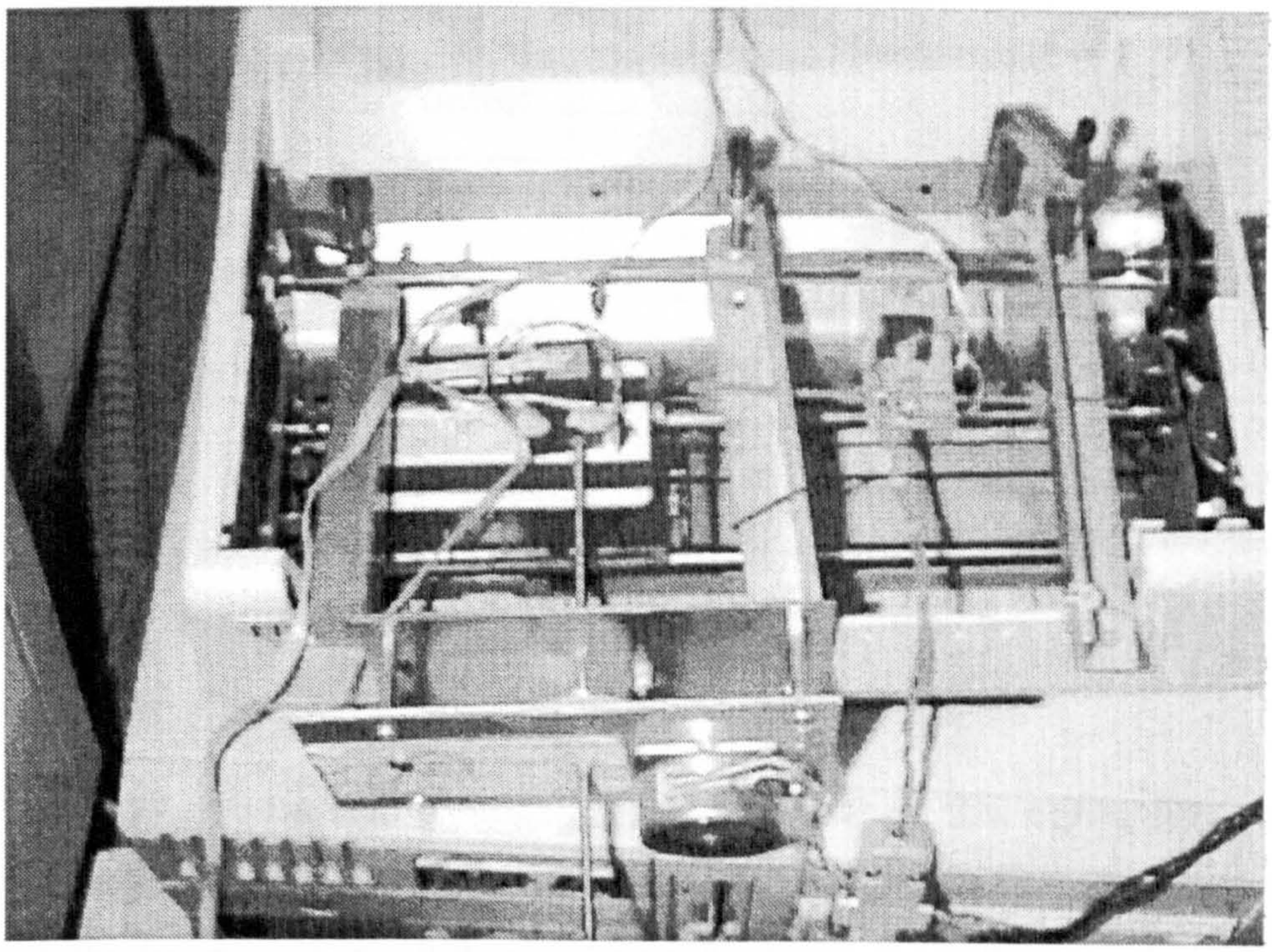


Figure 2.3: Photograph of printer showing lateral head motor and gearbox (bottom), the write head and I.R. sensor (centre left) and an inductive read head mounted for testing (centre right).

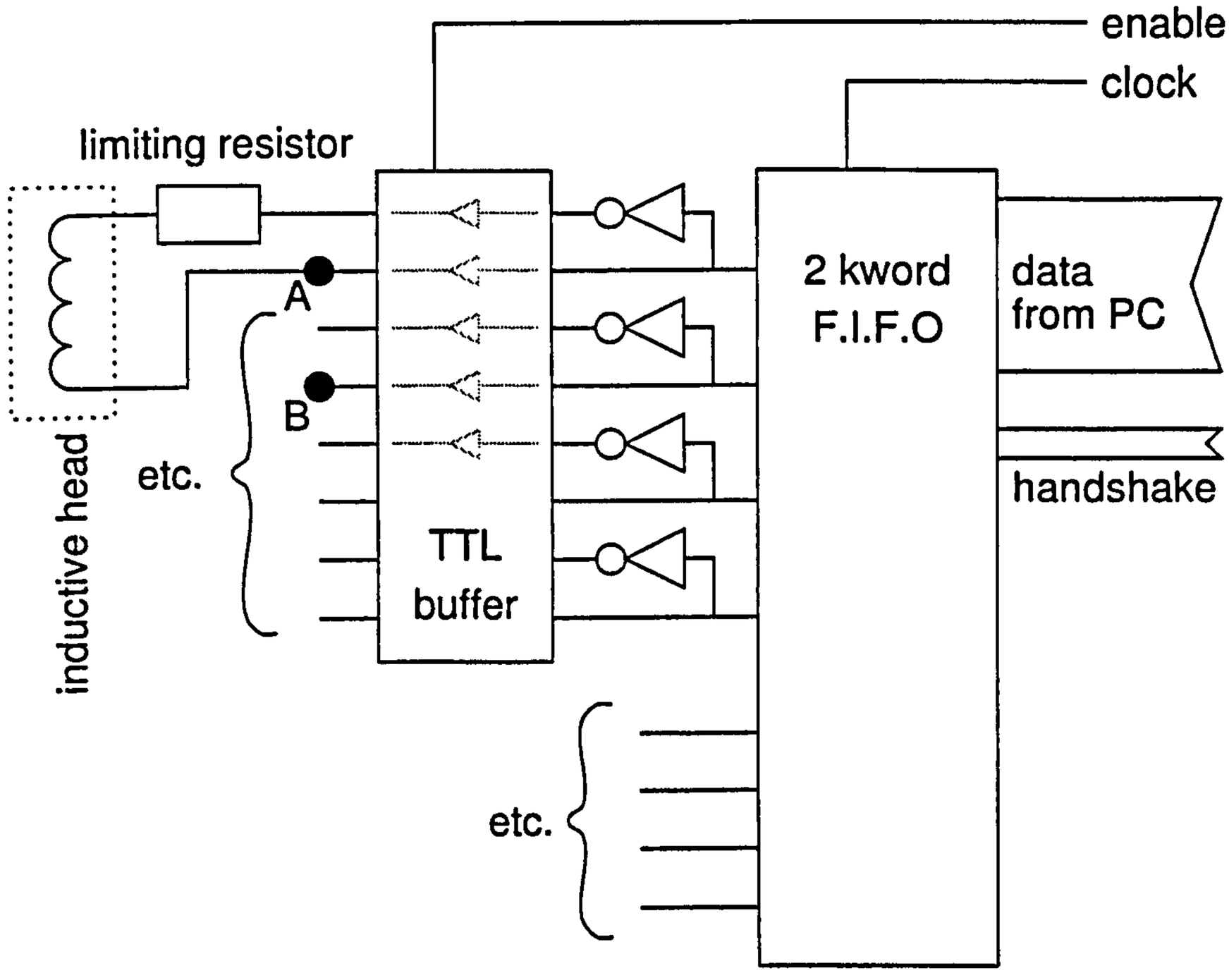


Figure 2.4: Schematic of write circuitry.

which is clocked at a stable programmable frequency through a FIFO buffer and then sent to a multi-track head. Current can flow backwards or forwards through the head coils, or all the tracks can be turned off (figure 2.4)

By taking two track signals together, current can be individually controlled to flow backwards, forwards or not at all through up to 4 heads. This allows a pulse-write system (see the results in section 5.1.3, page 99, for why this is useful), and a small power amplifier (figure 2.5) was built to increase the write current to a single, wide gap, head.

2.2.3 Write system head

Three types of write head have been tried with stripe cards.

- A four track compact cassette audio head.

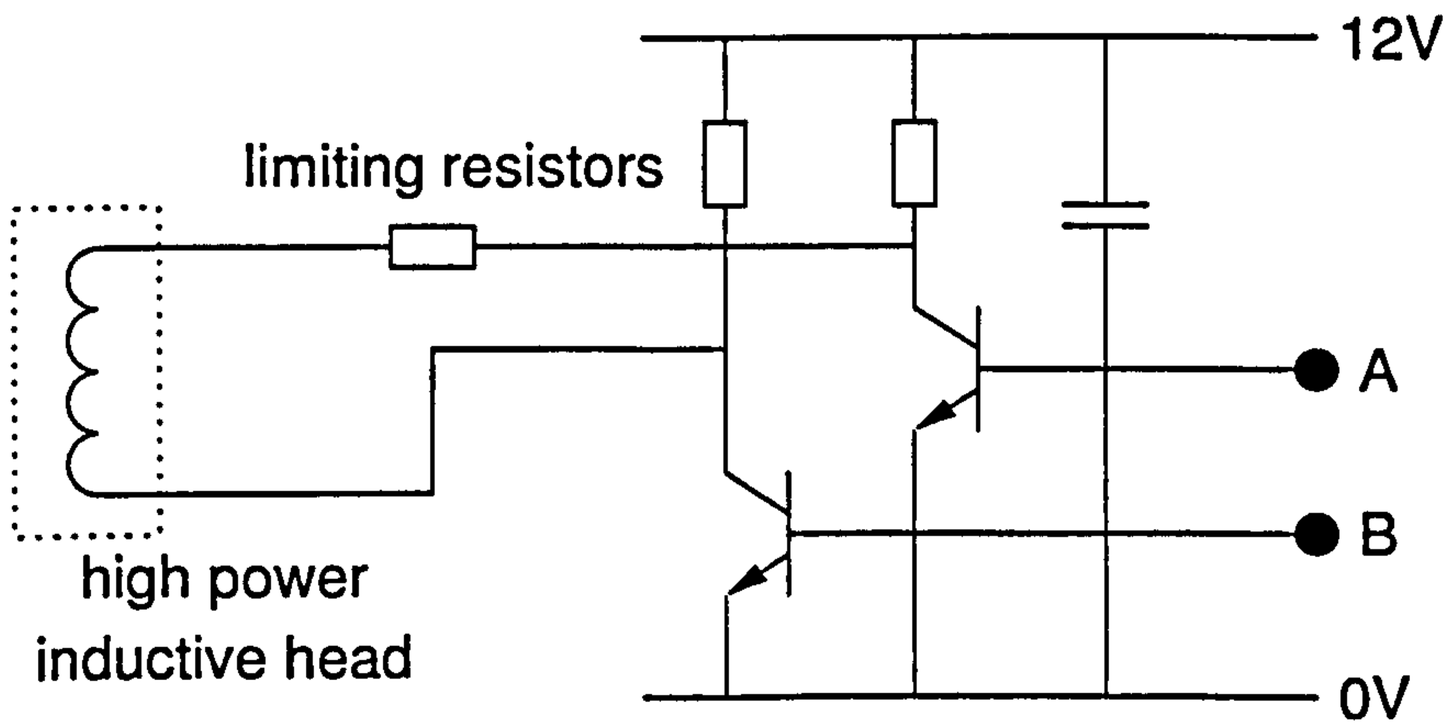


Figure 2.5: Schematic for high power write head amplifier.

- A single track, wide gap, high current specialized card head.
- A nine track, thin film Digital Compact Cassette (DCC) head.

DCC: Digital Compact Cassette

Of these, the audio head gave poor results on replay (figure 2.6, top trace). The thin film digital head gave no replay signal. These two results can both be related back to the protective layer on the surface of the magnetic medium (section 1.2.1) and the head gap (approximately 3 µm and 0.7 µm [57, 106] respectively).

The specialized write head gave far superior results to the audio head in terms of amplitude, but (as will be seen later, section 5.6.2, page 139) also caused problems with jitter. Microscopy suggests that the head gap is around 15 µm for this head, although the exact figure is not known. The lower trace in figure 2.6 shows the replay signal from a track recorded with this head.

2.2.4 Summary of write mechanism

It must be recognized that the card writer set-up was a one off, designed specifically for the needs of this project. A commercial writing plant would almost certainly use parallel write heads (or separate write heads) so that writing speed can be very high. A commercial concern would also place a large emphasis on

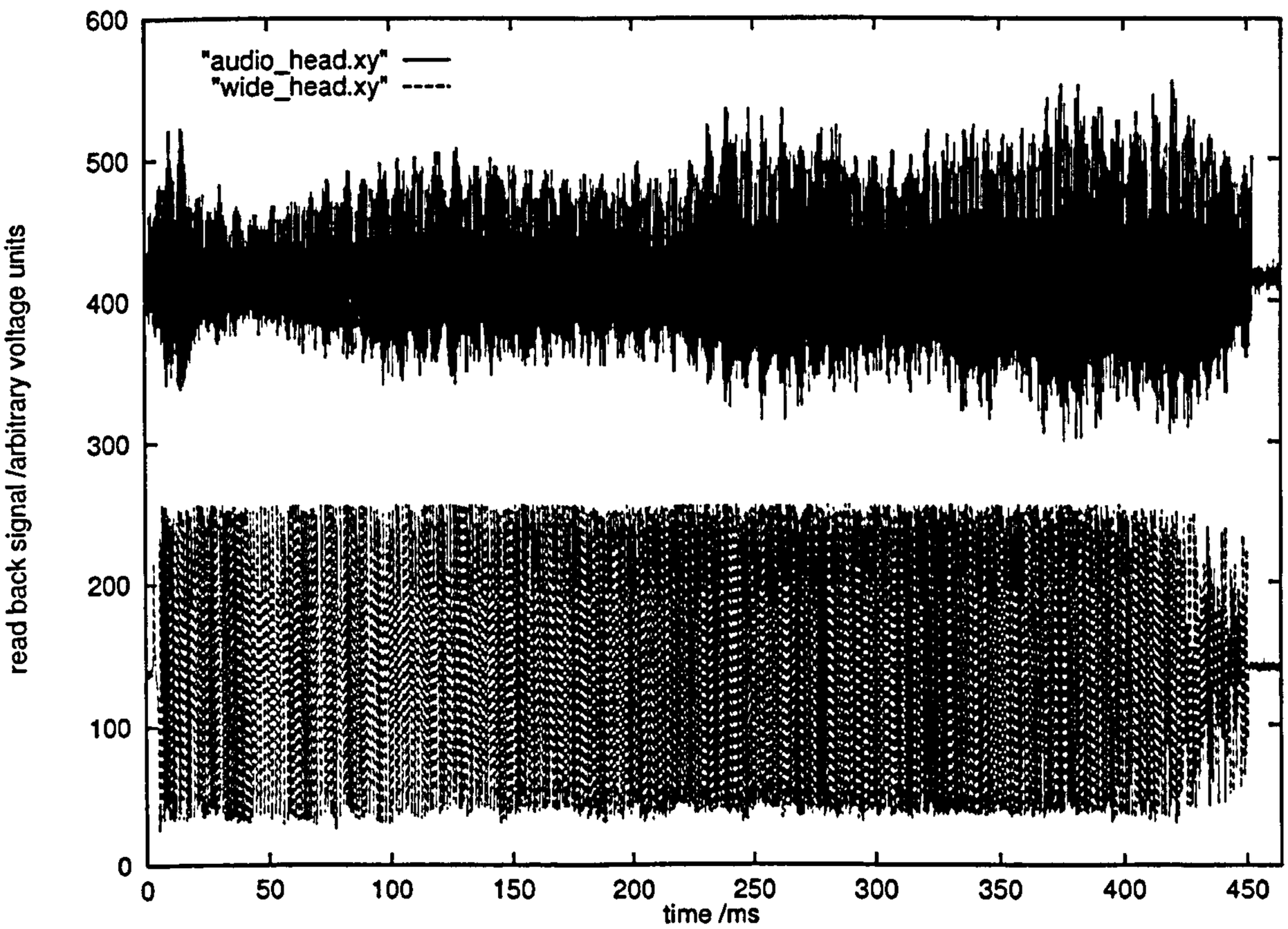


Figure 2.6: Replay signals from tracks recorded with the audio head (~ 3 µm head gap, top) and specialized card head (~ 15 µm head gap, bottom). The upper signal shows large amplitude swings due to the variation in protective layer thickness. Both signals were read with a DCC MR head.

head design, something that was not possible in this project. However, the final aim of the write mechanism is just to write magnetic patterns to the medium: the product from either this experimental apparatus or a large scale plant can be made identical.

2.3 Read mechanism

There are also three parts to the read mechanism: magnetics, electronics and mechanics. However, the hardware is not as complex as that of the write side.

Broadly, a hand swiped card is moved past a multi-track head. Each track signal from the head is sampled, digitized and relayed to a computer where it is stored for later analysis and decoding.

2.3.1 Read mechanics: the swipe card reader

Although it is possible to mount a read head adjacent to the write head in the card writer for test purposes, normally the head is simply positioned in a straightforward hand swipe guide. This is a slot that the card moves through and is shown diagrammatically in figure 2.7 and photographically in figure 2.8.

Precise alignment in yaw, azimuth and vertical position is impossible with such a device. In addition, the speed of the swipe is unpredictable and can vary considerably. However, it remains the most popular and inexpensive of the common card reader mechanisms, so improving on this imperfect equipment is undesirable. Section 1.2.5 (page 28) examined some of the read properties of standard cards in this reader.

2.3.2 Read heads: inductive and MR

Three factors affect the choice of head.

- mounting constraints
- signal response

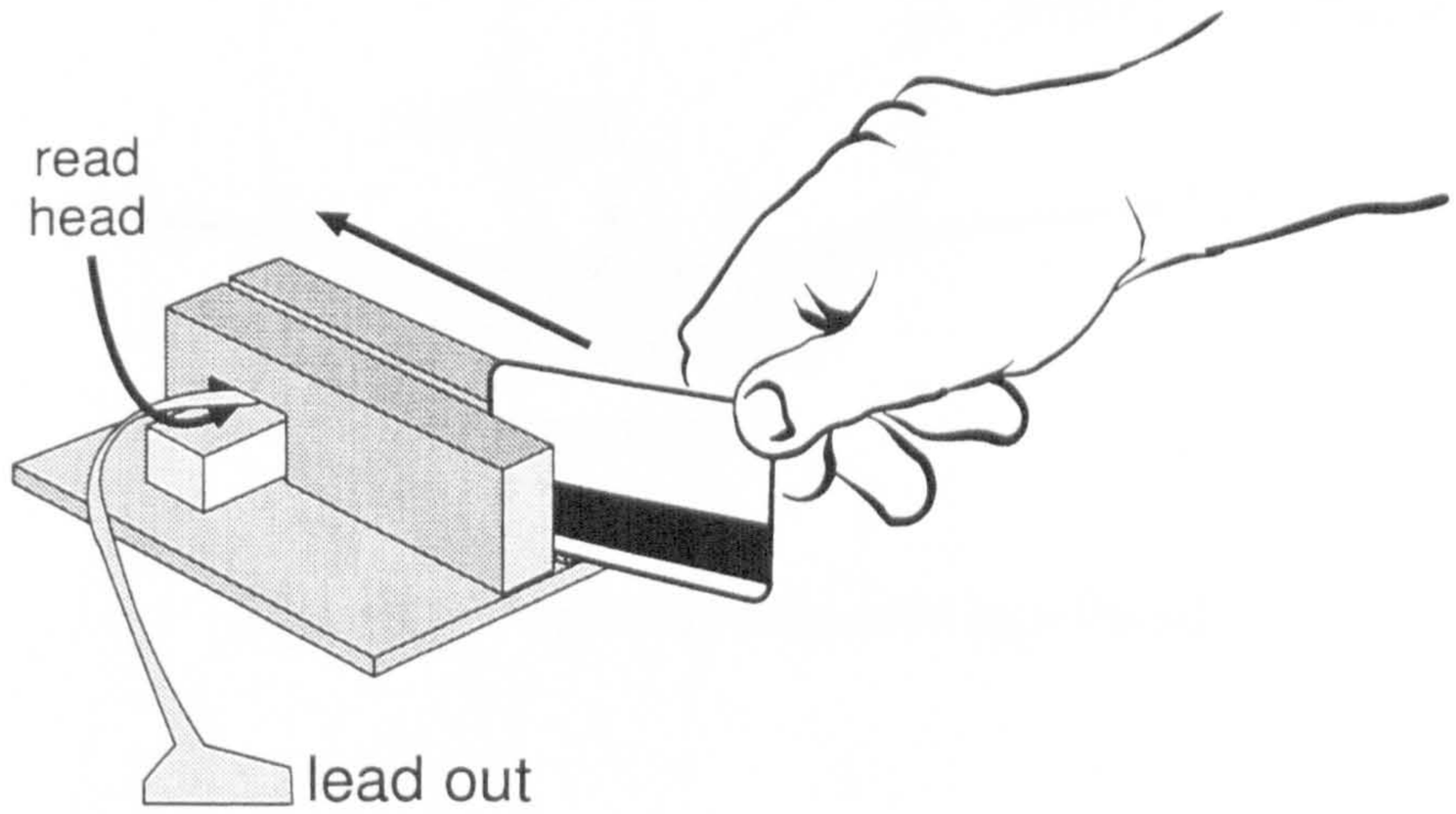


Figure 2.7: Schematic of slotted swipe card reader.

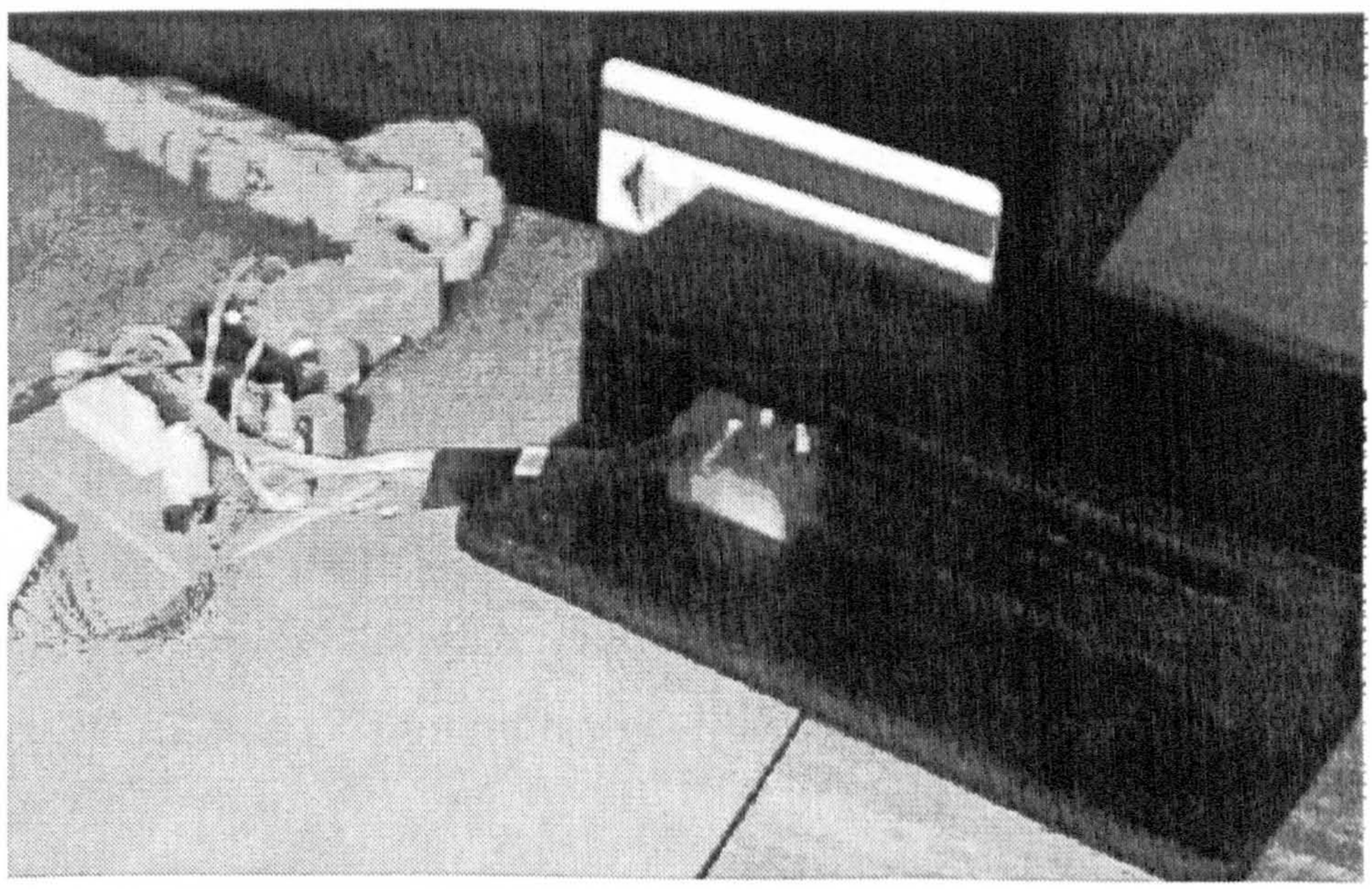


Figure 2.8: Photograph of swipe card reader with DCC head.

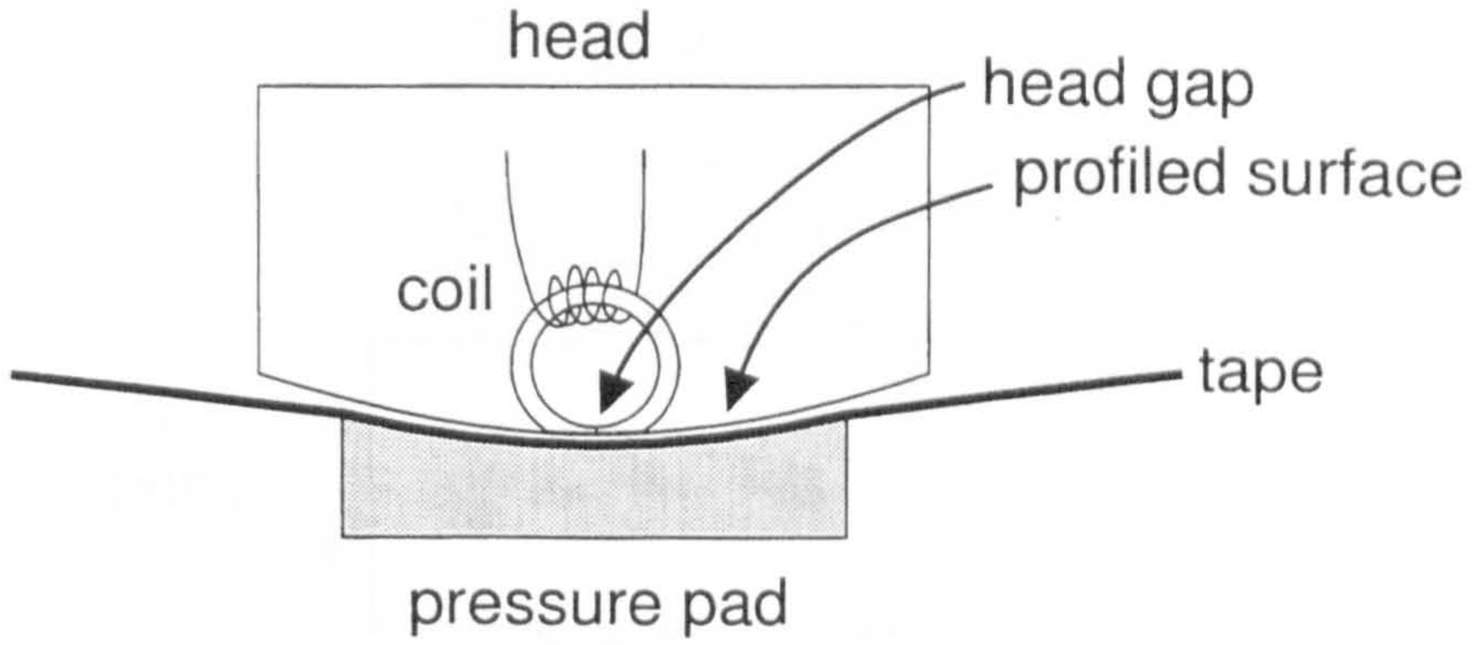


Figure 2.9: Profile of inductive tape head.

- durability

In any commercial system only the last two points are of major concern, but in a small scale set up, the first is by far the most challenging.

Two very different heads were used, both however were designed for compact cassette: a flexible tape medium. The head must be mounted in such a way that the gap is in contact with (or very close to) the magnetic medium. Tape heads are not ideal for cards as the surface of the head is profiled (figure 2.9) to allow the tape to wrap around and be pressed against the head surface. Plastic cards are not flexible on that scale, so a lot of effort went into ensuring that the active part of the head was touching the card, including a set of re-lapping experiments [unpublished] that showed that flattening the head did *not* improve the quality of contact over careful positioning.

Ultimately, a precise fixed mounting (figure 2.10) was found to be adequate. Nevertheless, commercial readers mount their heads flexibly with yawing ability (figure 1.15, page 27) to improve the read reliability when a card is poorly swiped.

Inductive audio cassette head

A standard 4-track audio compact cassette head was evaluated and used. As with all inductive heads, this gave back a voltage proportional to the rate of

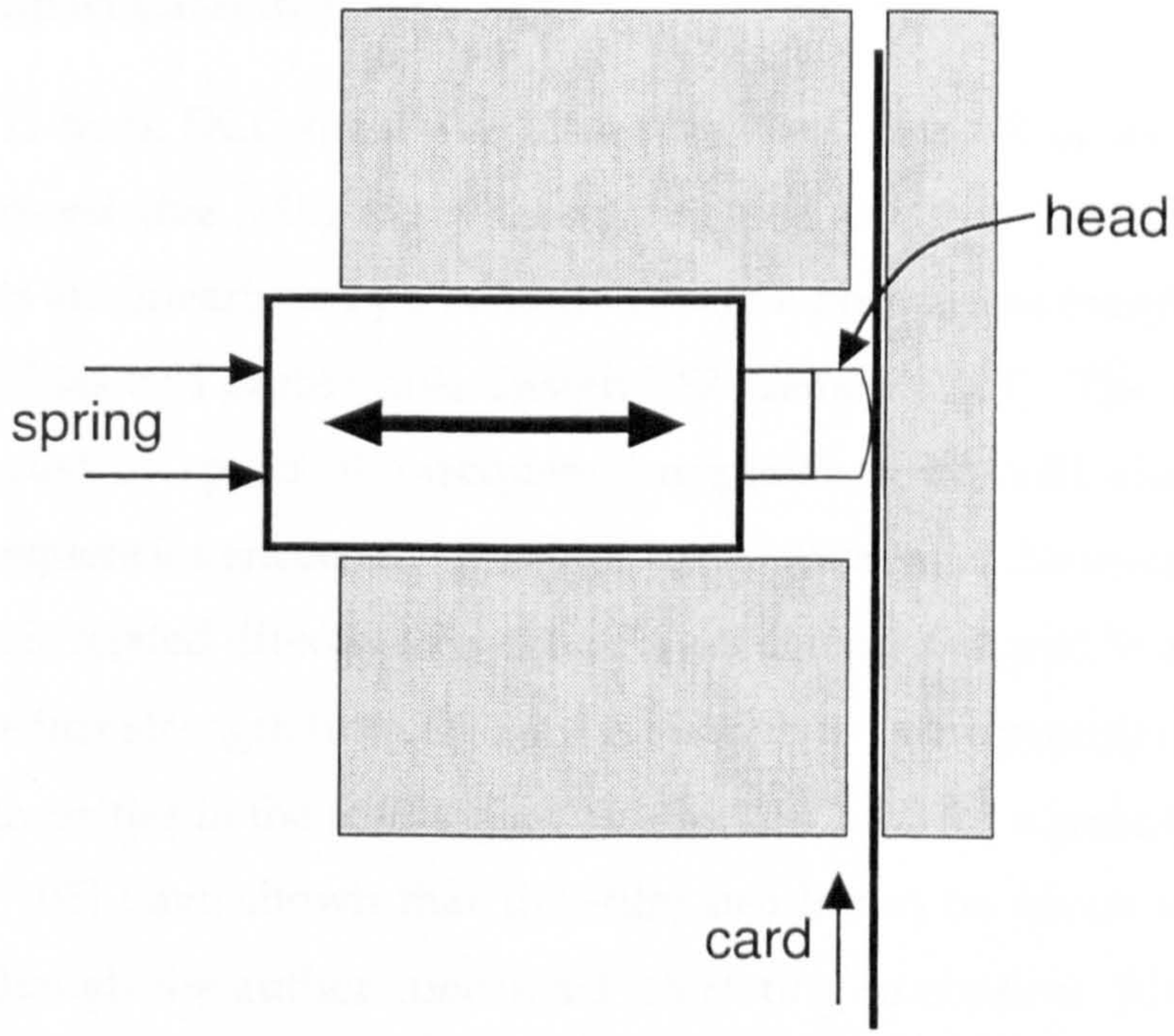


Figure 2.10: Rigid mounting found to be suitable for credit card reading.

magnetic flux change, hence proportional to the speed of the card. This can give serious problems with fixed threshold detectors, but is less of a problem with digitizing detectors. However, in practice, the changing signal level does still complicate the decoding process (a lower signal level implies more quantizing noise and adaptive filters need time to re-adjust; high signal levels can lead to clipping which, if asymmetric, can severely disrupt adaptive linear filters). The measured signal to noise ratio (SNR), nevertheless, was good at most speeds. In commercial systems, automatic gain controls (AGC) are often shunned in favour of low digital thresholds on grounds of cost. No AGC facility was built into the digitizer.

SNR: signal to noise ratio

AGC: automatic gain control

Durability was not studied. It is interesting to note that after grinding the surface of the head over 150 times with glass paper affixed to a credit card (so that the head was visibly worn) the read signal was not significantly degraded.

Digital Compact Cassette (DCC) head

MR: magneto-resistance

A modern 11-track DCC head was also evaluated, using 8 of its tracks. This is a magneto-resistive (MR) effect head fabricated as a thin-film structure. The MR elements are linearized by a combination of a permanent magnet bias, electromagnetic bias and barber-pole design MR elements [78]. The signal levels are not affected by speed (the frequency response of the MR element far exceeds the frequencies encountered in this application and the resistance of the MR element is related directly to magnetic flux density). Signal levels are good, although the flux strength from the card is enough to cause prominent and awkward non-linearities in the read signal (see section 1.1.8 for a partial solution).

Studies [105] have shown that thin-film heads can be given very durable coatings although the author conducted no testing to confirm this. No signal degradation has been encountered with the (single) DCC head used for this project so far.

2.3.3 Read electronics: amplifiers and digitizer

I/O: input/output

Sampling the signals from 8 heads presents practical problems in computer I/O transfers as the data rate can be up to 0.5 MByte/s while still having to maintain an accurate sampling period. A direct memory access (DMA) interface was designed and built to cope with these high data transfer rates; a small (16 byte) FIFO was still required to cover the computer's interrupt latency. Signals from a head are passed through a high-gain amplifier directly into an 8-channel ADC. The digital data then flows, via the FIFO, into a DMA port of the analyzing computer. The circuit diagrams are again presented in appendix C.

FIFO: first in—first out
ADC: analogue to digital converter

2.3.4 Summary of read system

The entire read system is a slot for the card to travel through, a magnetic read head (that can read several tracks in parallel) and a digitizer. This represents the hardware cost of the high density card system. The additional processes required to extract the data from the card are software based, and are performed

in the domain of digital signal processing. The highest cost component is the read head (around £30 in small quantities, probably falling to a quarter of that price for large volumes). The second most costly component is the ADC (at around £8). The computer doing the processing (an inexpensive 486 or equivalent) is already present at most points of sale, and does not figure in the costings.

2.4 Apparatus and track density

The construction of a high density credit card requires the cooperation of several facets of communication technology. Areal density is the product of track density and linear density. The track density probably presents the most scope for further increases, but it is also the least flexible, given the constraints of card swiping. Lineal density can only provide a limited density increase, but there is a great deal of scope for achieving that.

Track density has not been examined in depth during this project, but the subject is covered briefly in this section. The track layout is governed by the availability of heads.

2.4.1 Track density limits

If track density could achieve the same resolution as linear density, with a very large number of heads (several tens) and a two-dimensional processing routine [95] to decode the entire magnetic stripe as a single *image* of magnetized regions, then 400 tracks per inch (tpi) could, theoretically, be decoded. Under these conditions, noise would be the dominant limit to track density. However this 50-fold density increase (about 30 kB on a card) over a standard card cannot be achieved in practice; not least because such heads do not exist at present.

tpi: tracks per inch

Given that the track density limit is in fact the width of a read head, the factors that dominate practical limits will now be touched upon.

2.4.2 Magnetic read head

DCC: Digital Compact Cassette

The most relevant limit in this project was the availability of read heads. The Philips DCC head has a track pitch of 195 µm (130 tpi) (figure 2.11). The read elements are much smaller than the track pitch at 70 µm (1/360 inch). This was a design feature so that there would be fairly minimal inter-track interference when used for digital tape recording and it does not pose any particular difficulties in the card environment.

2.4.3 Track interference on reading

The second limit is due to track misregistration. As the card passes through a hand swipe reader (and mechanical readers and writers to a lesser extent) the transverse head position varies. In tape and disk systems this is compensated or limited by mechanical devices: from simple guide rails to sophisticated servo systems.

The amount of card misregistration is largely attributable to the way the card is swiped through the reader. A steady swipe can give variances of less than a few hundred microns. A hasty pull might give several millimetres of variation. Although no scheme can cope with the situation where the magnetic medium leaves the head completely, it is at least possible to correct a few track-widths of misregistration in software [7]. The mixture of signals read from a head overlapping two tracks is predictable and the individual signals can be separated. Figure 2.12 shows the measured quantities of signal amplitude from four inductive heads as laid out in figure 2.11, obtained from the setup shown in figure 2.3. The track signal strengths rise fairly linearly and have a flattened top. This is what one would expect from a wide head passing over a narrower track (and vice versa). The results were obtained by writing one track at a high frequency, surrounded by tracks written at a lower frequency, the signal strength recorded is the proportion of high to low frequency content in the read-back signal.

For practical purposes, an unqualified track width of 0.4 mm was sufficient to make inter-track interference avoidable, as two of the DCC read elements can

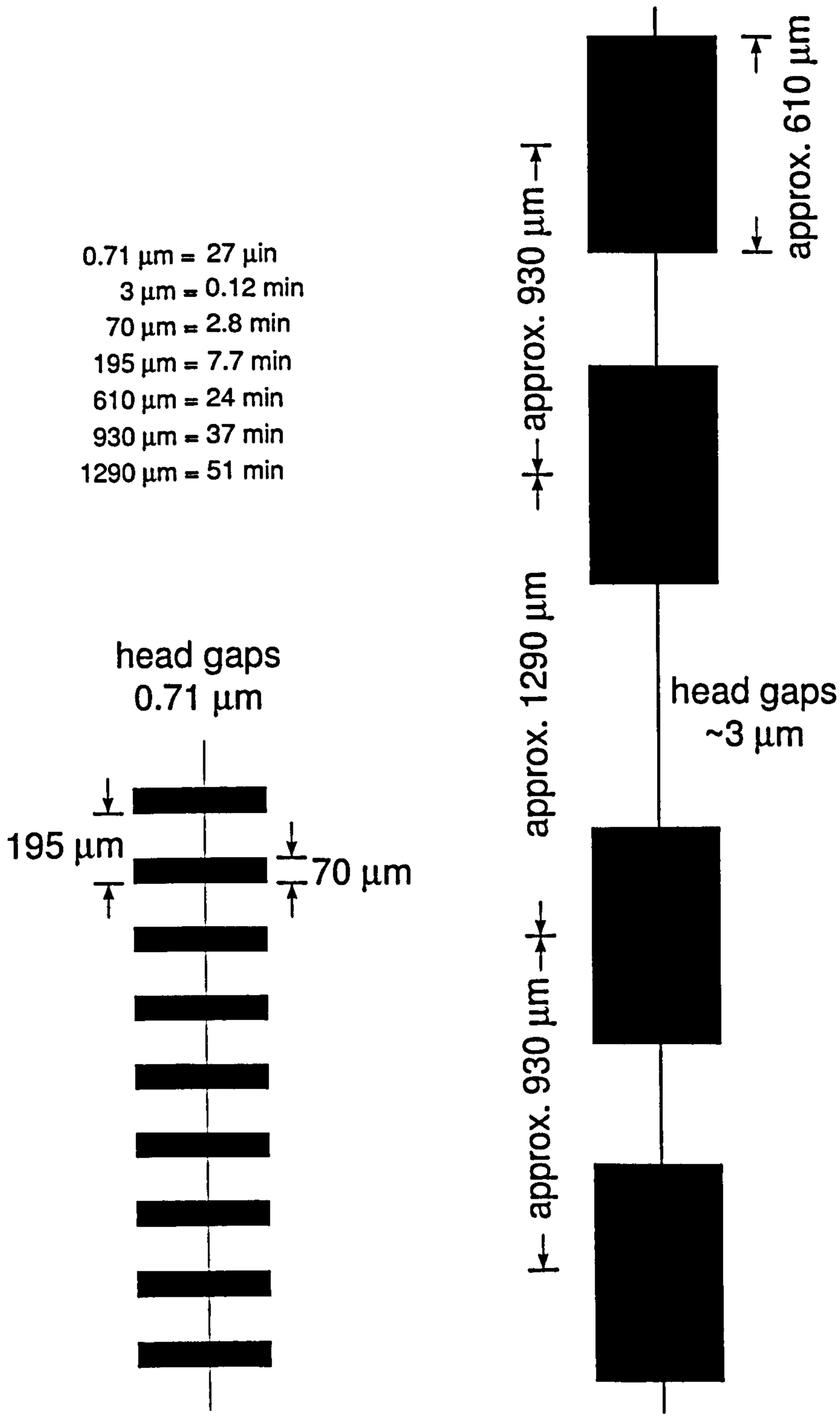


Figure 2.11: Layout of read elements on a DCC head (left) and an inductive audio tape head (right).

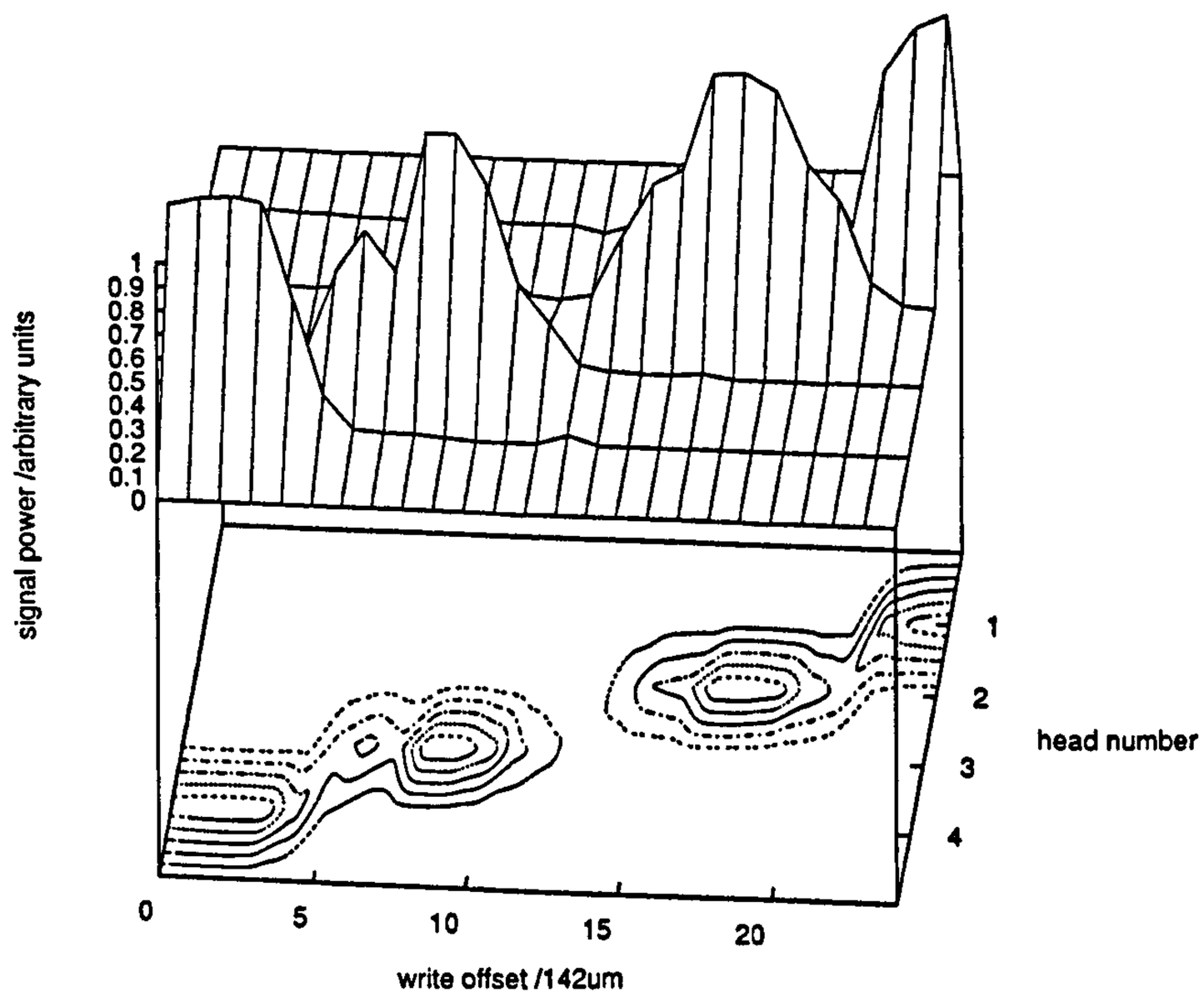


Figure 2.12: Measures inter-track interference with a compact cassette head. A track 800 µm wide was read at different lateral offsets. Four inductive heads of width ~610 µm, the layout is shown in figure 2.11-(right).

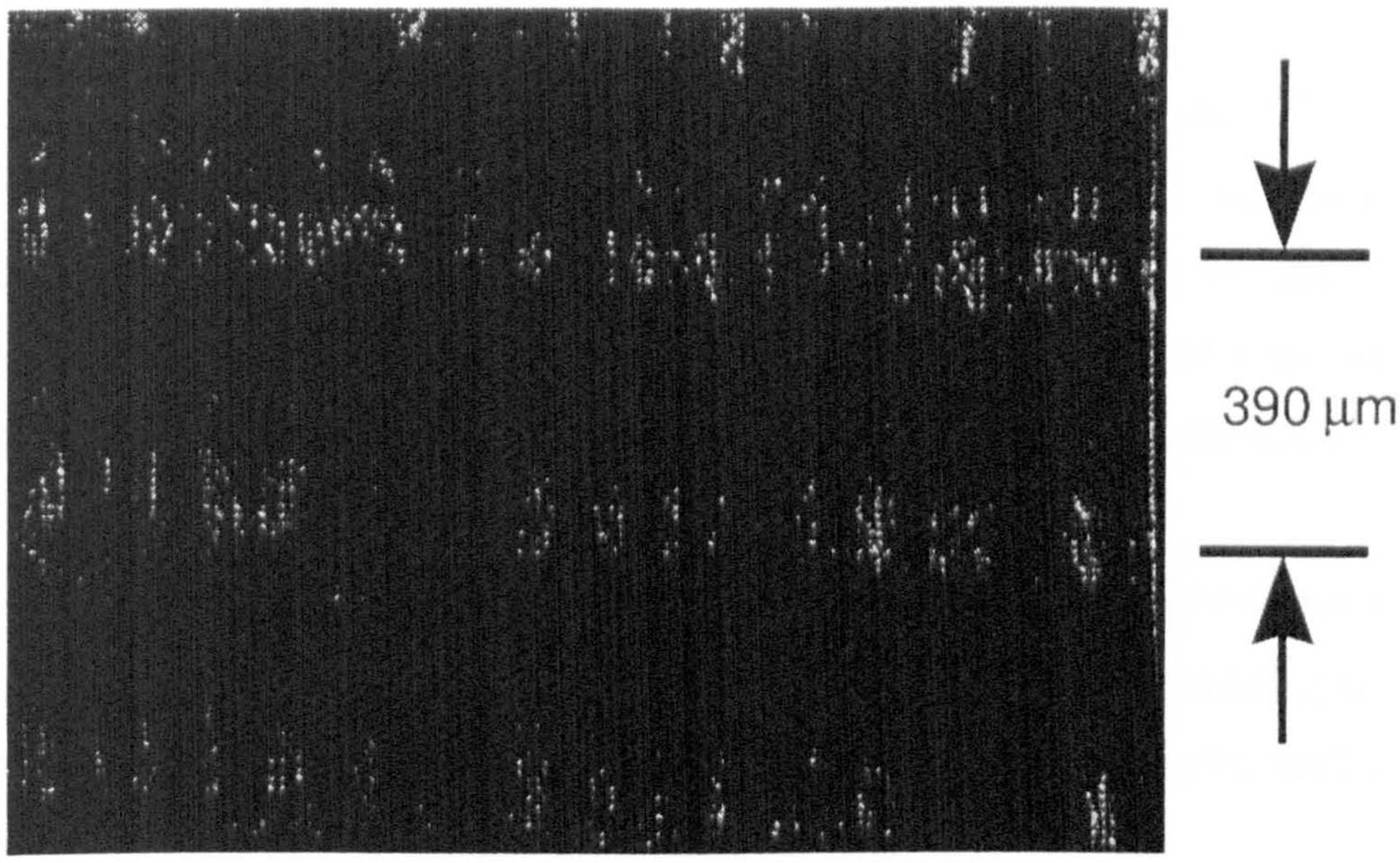


Figure 2.13: Ferrofluid photograph of card surface showing track edge erasure (some longitudinal and transverse scratches are also visible on close inspection).

read each track, and the software can merely switch between the best signals in a manner similar to multi-head analogue video recorders, utilizing the narrow width of the DCC read heads to give a clean signal.

2.4.4 Track fringing on writing

The third limit is what is achievable when writing data. Every write head has a field that extends laterally beyond the gap. The wider the gap the larger this fringing field will be. It was found that with the credit-card writing head these fringe fields precluded writing tracks of less than about 200 µm pitch (see figure 2.13). This is a head problem, and can be solved by redesigning the write head.

In fact this is a general problem for single and multi-track recorders: in the research lab hard disk write heads have been reprofiled at the gap edges with ion beams to reduce near track overwrite [36].

2.4.5 Practical track options

Although no extensive track density investigation has been carried out, it is possible to make some comments on a sensible way to lay down tracks on a high density credit card. The following track layouts all use the ISO track 3 to carry high density information. This leaves tracks 1 and 2 free to carry the standard credit card information—an enormous boon in compatibility although a great loss in data capacity.

A system with a well profiled write head and inter-track interference compensation could achieve the layout shown in figure 2.14. By repeating the edge tracks, the heads see the tracks as 'quasi-tubular' and no tracks will be lost when up to 390 µm of track misregistration is present.

This project has achieved half of this track density, and thus ignored the problems of write fringing and inter-track interference. With the track structure shown in figure 2.15 between 0.3 and 1.5 kB can be stored in the track 3 region, with varying degrees of reliability.

2.5 Software interface

All software work has been developed on a Unix-based IBM compatible PC running the Linux operating system, shown in figure 2.16.

Device drivers were written for the write head and read heads so that both systems appear simply as stream character devices with special features controlled via the Unix `ioctl()` interface (clock rate, number of channels, write amplifier activation, infra-red sensors, etc.).

A character device driver was also developed for the write mechanics. The parallel port and printer were enhanced to allow the paper feed motor to operate backwards *and* forwards. In addition, a software pipe converted high-level card writer commands ('`XTO 40`', '`FAST`' and so on) into raw printer commands on the parallel device. This allowed a great deal of the development work on the card writer to be written and maintained in high-level shell scripts.

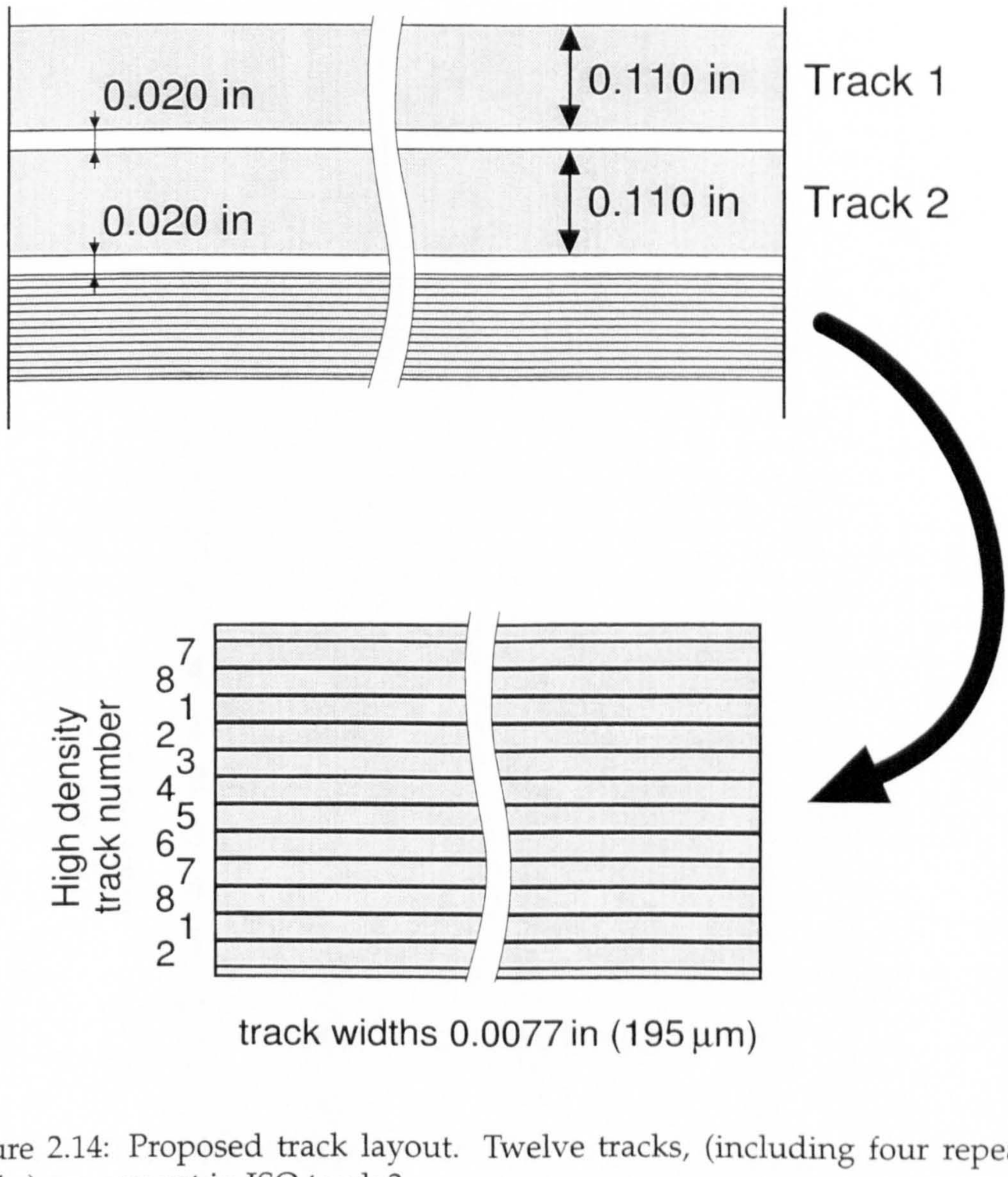


Figure 2.14: Proposed track layout. Twelve tracks, (including four repeated tracks) are present in ISO track 3.

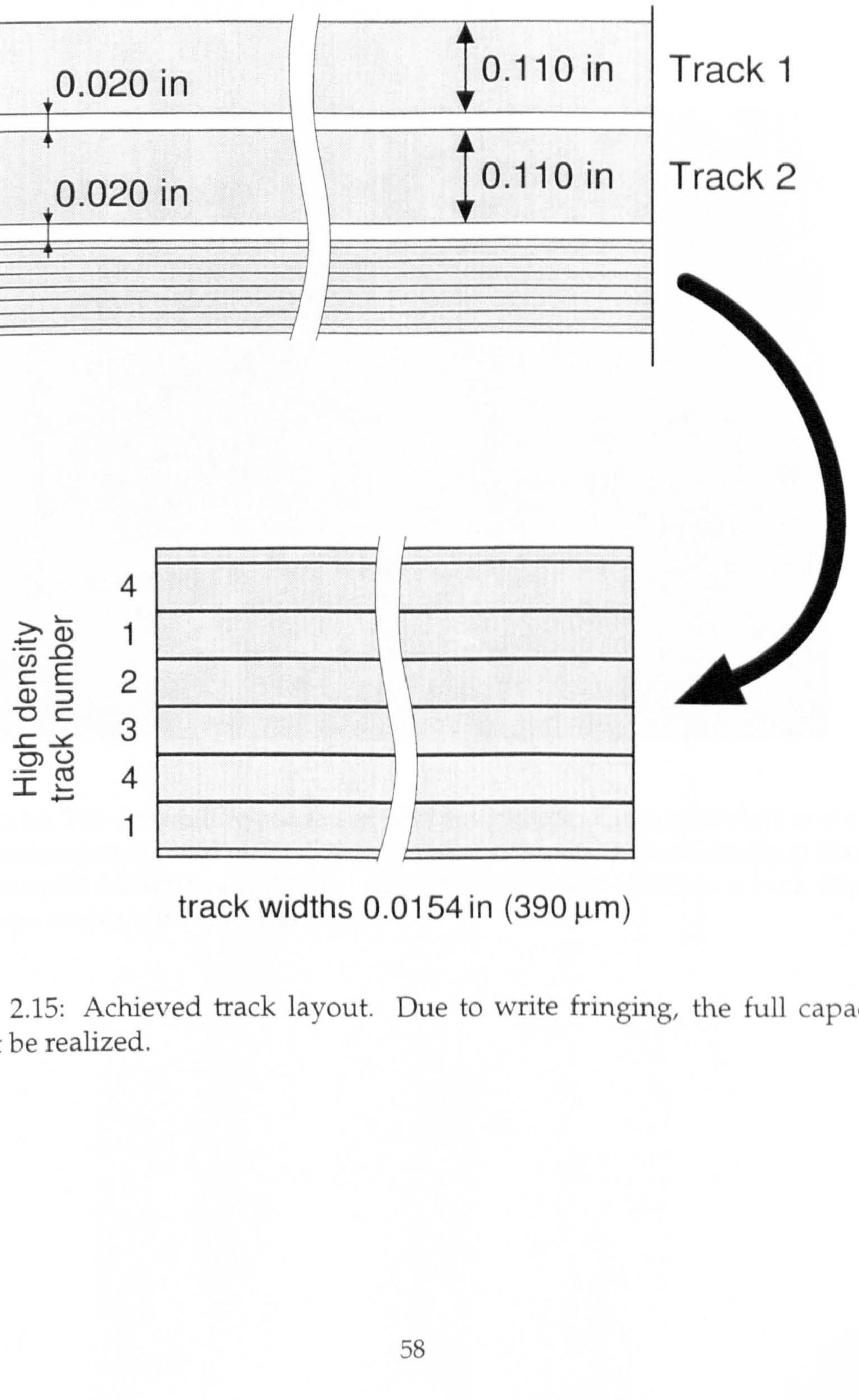


Figure 2.15: Achieved track layout. Due to write fringing, the full capacity cannot be realized.

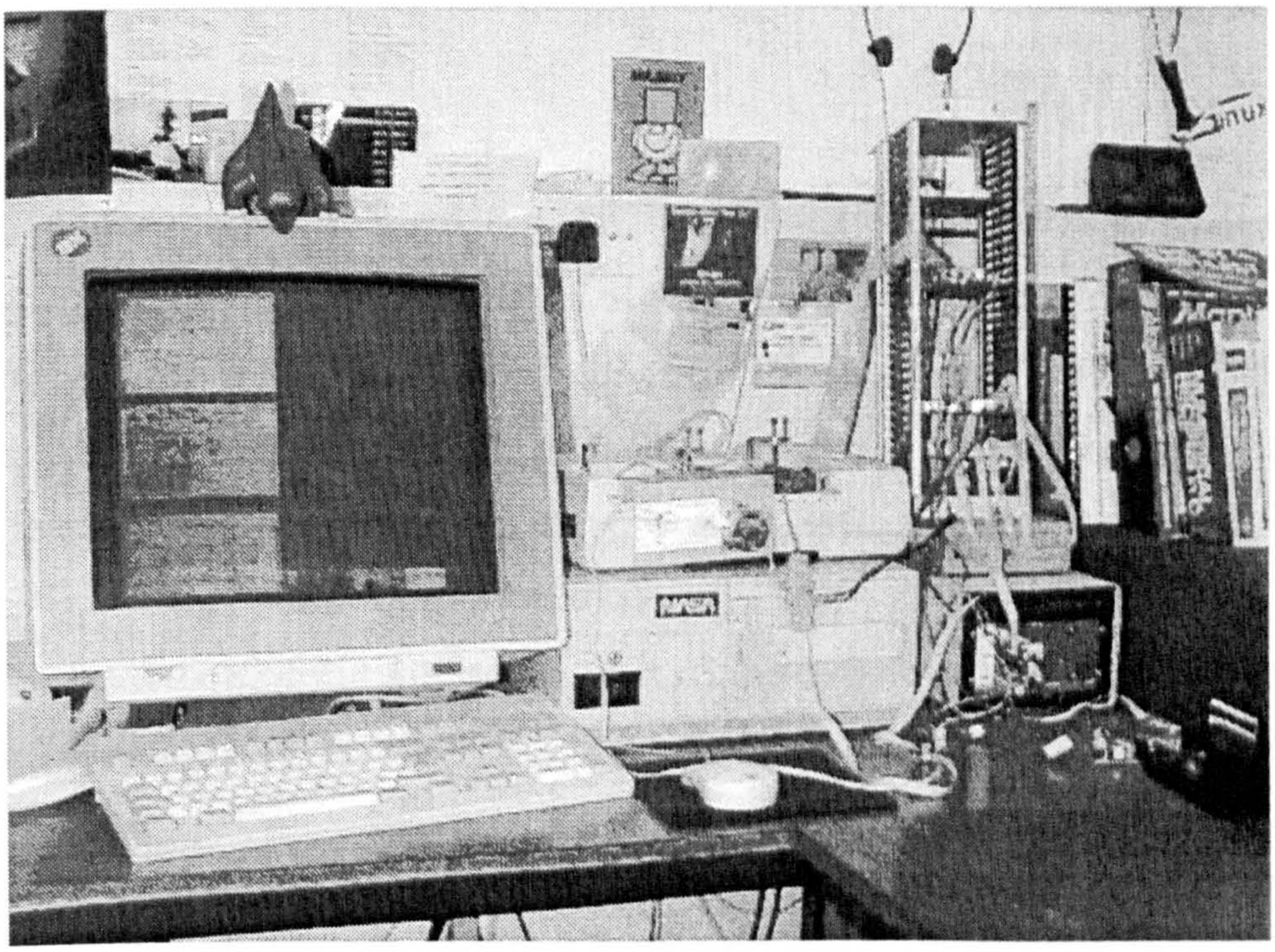


Figure 2.16: The controlling computer and equipment. Computer (left and centre), printer set up as card writer (centre, above computer), power supply (lower right), circuits for write amplifiers, read amplifiers and ADC in a rack (right) and swipe reader with card (far right).

Most of the controlling and processing software was written in the form of small Unix pipe filters. For example, programs to convert binary streams into ASCII binary, several programs that convert ASCII binary into (d, k) codes (and the respective decoders). Another pipe could convert coded streams into pulsed or normal output for the write head stream. The read head stream, FIR filters, detectors, decoders and reconstruction programs also worked in the same way. Some of the programs are reproduced in appendix D.

2.6 Summary

A system for reading and writing cards was constructed that was also capable of measuring the basic charateristics of the channel. The card is carried on a flat bed underneath a lateral head positioner. The head current is digital and capable of writing multiple tracks simultaneously at a programmable clock rate. The card reader is fundamentally a multi-channel digitizer placing a digital capture of the head output into computer memory.

The track layout for this work was dictated by the availability of replay heads. Although two of the heads were designed for audio cassette, there was no particular problem when reading from the card medium. However, writing cards required a special wide-gap head to be used.

Chapter 3

Coding

NRZ: *non-return to zero*

Raw binary data streams (NRZ, non-return to zero) are rarely suitable for direct broadcast onto a channel. On the other hand a binary stream written with circles and lines for zero and one, e.g.,

```
0100101101101011010000001011110111010111,
```

is enormously redundant when read by humans, and would not be particularly prone to errors when sent 'through' a paper channel. However, it is also not particularly efficient in terms of space. The same sequence could be written

▪ ▪▬▪▬▪ ▪▬▬▪▬

which is more efficient (it takes up less space on the page) but is also more difficult to decode correctly. (Are there 5 or 6 '0's in a row?)

The former representation coded the binary digits 0 and 1 in the shapes '0' and '1' while the latter used the shapes ' ' and '▪'. The code used to represent data and the detection method used to retrieve the data are intimately related.

Digital magnetic recorders can only use the ▪ method. In addition, the ▪s are fuzzy—not black and white but shades of grey. To retrieve information from a magnetic channel the data must be coded, detected and decoded. The scheme should also use the channel efficiently.

The information depicted in figure 1.14 is written to track 2 of a credit card as below (real size, a dark area represents magnetization in one direction, a blank area magnetization in the other).

3.1 Properties of modulation codes

The first line of attack is to break up long runs of ▬ because it is difficult for the decoder to count the number of individual ▪s. In some situations, it is also advantageous to remove very small runs because they may be missed completely. A run length limited (RLL) code does exactly this.

RLL: run length limited

There have been many RLL codes designed for different systems, with different intrinsic characteristics. Some of the relevant properties are described below.

3.1.1 Run length constraints

RLL codes are often classified as (d, k) codes: $d + 1$ represents the minimum allowed run length of a symbol and $k + 1$ represents the maximum allowed number of identical symbols in a row ($k > d$). The designer of a code must decide how to represent information so that the (d, k) constraints are satisfied. A smaller k and larger d makes detection easier.

The (d, k) constraint reduces the information carrying capacity of the code. The 'capacity' of a (d, k) code is the measure of how much information that code can carry, relative to $(d, k) = (0, \infty)$.

3.1.2 Charge constraint

Some detection systems (most notably those used in optical recording applications or that must contend with small signal levels) require that the positive signal is balanced against the negative signal so that the zero point can be found accurately—usually for clocking purposes. If this is the case then it is imperative that the code does not upset this calculation, implying that some sort of maximum charge imbalance must be enforced. The running digital sum (RDS) is a calculation of the charge imbalance so far and is linear to (long term) time. DC-free codes have an average RDS that is zero, and a maximum instantaneous RDS that is labeled the digital sum variation (DSV). A lower RDS or DSV eases

RDS: running digital sum

DSV: digital sum variation

the complexity of signal amplifiers and detectors.

3.1.3 Codes in multi-track systems

The traditional way to cope with inter-track interference is to leave a space, a 'guard band', between tracks. The size of guard band directly affects areal efficiency, however, so it is highly desirable to reduce the size of this unused space*. Inter-track interference has long been recognized as a problem. Different methods exist to cancel the interference [7, 46] and to make the interference powerless [42].

In a system where tracks are overwritten, inaccuracies in the head position can lead to some data signals spilling into the guard band areas. Hence models of inter-track interference must allow for the old information in the guard bands, as well as information from neighbouring tracks, as its signal contribution is substantial (up to about 10% [93] of the signal level). To reduce this problem, systems often use a wide write head and a narrower read head, so that the detector sees reduced out-of-track signal levels.

If there are multiple tracks being read simultaneously, the k constraint can be relaxed slightly because some timing information can be derived from the other tracks—a two-dimensional coding system [24].

3.1.4 Timing window

While the detector is running, it must make decisions about where the centre of a symbol is. The amount of timing error the decoder is allowed before it will make mistakes is called the timing window, normally measured in terms of the symbol rate. A larger timing window is better.

The consequences of a timing error are normally a few bits in error. However, if the loss of bits also causes a loss in the timing clock recovery, the mistakes can be compounded. The resilience of codes to timing errors is treated in a section of its own in 3.2.

*In disk drives this space is utilized as part of the head servo alignment mechanism.

3.1.5 Transition density and efficiency

The efficiency of a code is perhaps its most immediately obvious property. However, the channel can often dictate code efficiency by the error rate more than the designer would like: it is unusual for the data throughput to be dominated by the rate or efficiency of the coding scheme. There are several common measures of code efficiency though. Some of these measures are formally stated in appendix B.

Most common coding schemes take m input bits and generate n coded output bits. The ratio of input to output bits is called the code 'rate'. Highly constrained channels may use codes running at rates as low as 0.5 (one data bit generates two coded bits). Highly developed hard disk channels run at rates of 8/9 (0.89) and are moving towards 16/17 (0.94). But with partial response channels and iterative detection schemes, the code rate is not particularly relevant to how efficient a code is.

The formal measure of code 'efficiency' is the ratio of the information capacity of the code to the information capacity of an uncoded (d, k) stream. If the efficiency is not approaching 100% then the code is underutilizing the channel.

The 'density ratio' of a code is a measure of the number of bits that are encoded in each transition. Since each transition contributes towards the required bandwidth of the channel, coding more bits per transition increases the channel utilization. However, higher density ratios can also decrease timing tolerances and increase sensitivity to noise.

3.2 Timing error resilience (TER)

The author has devised a new measure for codes. The timing error resilience (TER) is a measure of the ability of a code to recover from timing errors. The lower the value, the more powerful the code's self-correcting properties.

The TER value is the measured mean of the number of data bits in error before bit synchronization is re-established after a burst error during which the

timing phase was lost. (However, it is assumed that the frequency lock was not lost.) Because it is measured statistically, the results are somewhat dependent on the error recovery mechanisms of the decoding algorithm and it is not dependent solely on the code. It should be noted that these recovery algorithms, if they even exist, are not normally published. The approach the author took to developing self-recovering algorithms for standard codes is described below in section 3.2.2. Note also that the TER does *not* measure error propagation due to equalization or detection feedback mechanisms.

If a code cannot recover from a timing error, then it will eventually produce an infinite number of wrong bits. This is unacceptable for calculating the mean, so the maximum number of errors is truncated at 100 in the case of the TER-100 statistic. If half of the trials cause runaway errors, the TER-100 will give a figure of about 50 plus half the average number of bits in error when it *did* resynchronize.

The specific method of calculating the TER-100 values shown in tables 3.1 and 3.3 is given below.

3.2.1 Calculating the TER-100

Data streams of pseudo-random bits were generated and passed through the given coding algorithm to make new coded streams.

Each coded stream was then corrupted by a burst error of 20 random bits (so that the mean burst error length was 16 bits) at a random position. In addition, up to coded 8 bits were added or removed during the burst, so that synchronization was lost. This is shown diagrammatically in figure 3.1.

The coded streams were decoded to form corrupted data streams. The data bits in error were then counted by finding the first bit corrupted, finding the last bit corrupted (indexed against the original data stream) and taking the difference between the values. If the value was greater than 100, it was set to 100.

BIE: bits in error

The mean of these values is the average number of bits in error (BIE). To find the TER-100, the burst error length must be removed from the BIE. The mean

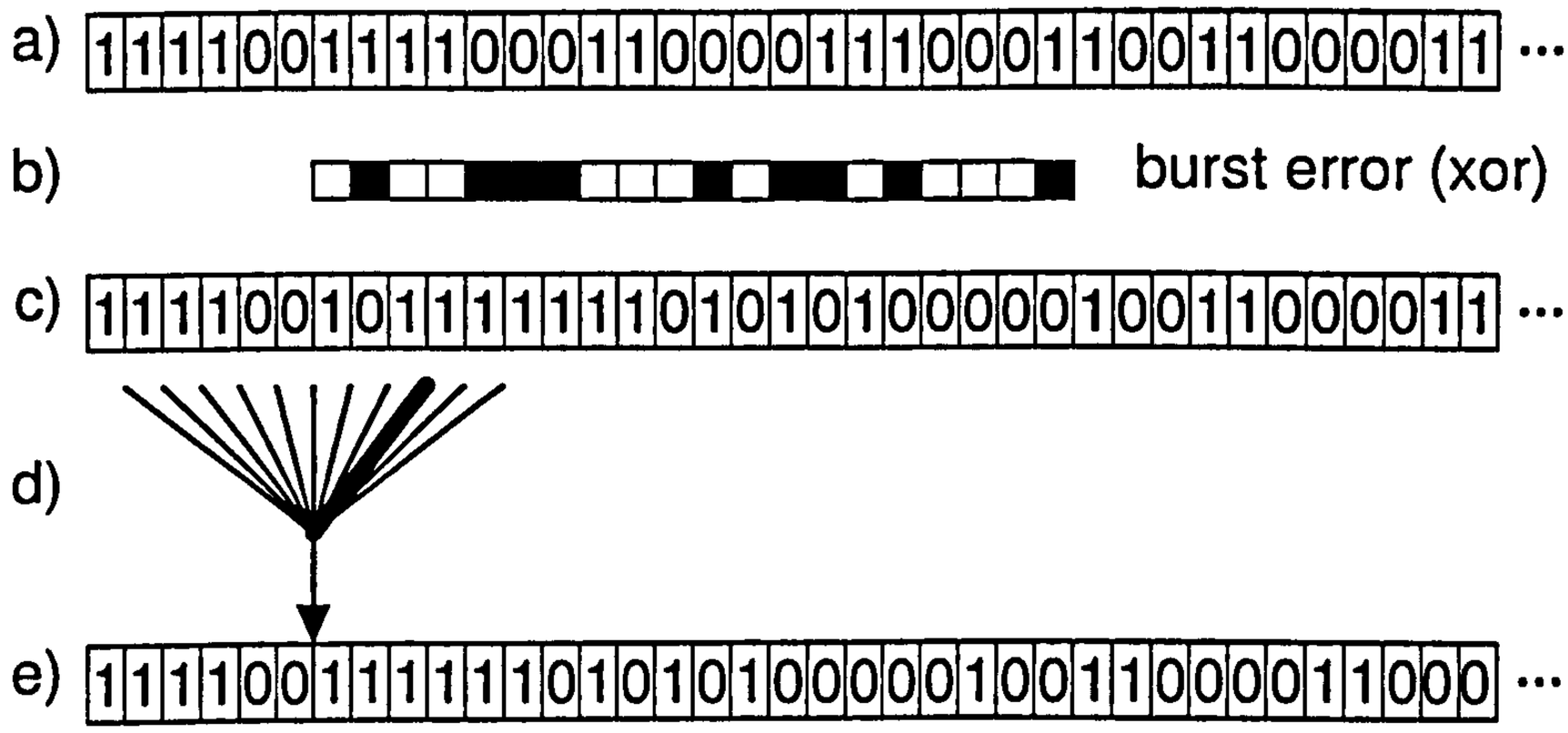


Figure 3.1: Scheme for introducing errors in the TER-100 test. a) coded binary data stream. b) burst error up to 20 bits inserted at a random stream position giving a corrupt coded stream, c). d) the stream is shifted up to 8 bits (up to 5 shifts are shown, with a shift of +3 being chosen) in either direction. When the decoding routine receives the corrupted and shifted stream, e), it is in an unknown internal state with no bit synchronization reference.

burst error length is simply 16 times the code rate. Subtracting this value from the BIE gives the TER-100.

When the TER-100 is small, it is also helpful to calculate the standard deviation of the BIE, as this gives an indication of how regularly the code can be corrected.

The results for several codes over 100 data streams are given in table 3.1.

3.2.2 A note on the decoding algorithms

It is sometimes very difficult to know how to correct a timing error in a code. The variable rate codes (section 3.3) and FM are self-synchronizing, but other codes, notably the block codes and codes with high efficiency, simply rely on statistical properties of random data to recover the bit clock.

The approach taken to designing the decoding algorithms for these codes was perhaps not optimal, but fairly robust.

If an error is detected (most codes can *detect* errors) then the current code

code	rate	BIE-100	TER-100	s.d.
NRZ (binary)	1.00	17.2	1.2	5.4
FM	0.50	9.4	1.4	3.4
MFM	0.50	12.2	4.2	7.5
ZM	0.50	54	46	73
RLL (1, 7)	0.67	91	81	85
RLL (2, 7)	0.50	76	68	80
3PM	0.50	74	66	77
BPPM-$1_{S=2}$	0.40	11.9	5.5	14
BPPM-$1_{S=4}$	0.27	10.2	5.9	5.7
BPPM-$2_{S=4}$	0.43	22.6	15.7	15
var. rate (1, 3)	0.52	10.8	2.5	3.6
var. rate (1, 4)	0.57	11.3	2.1	3.8

Table 3.1: Results from calculating the TER-100 for the codes in table 3.3

word was shifted, 1 code bit at a time, until it became a valid code word, and the decoder would go back to normal processing. In this way, given enough invalid sequences the code will always re-lock. Sometimes, however, these invalid sequences are rare, and not all codes have them (zero modulation (ZM) code for example, can detect an error, but cannot ever resynchronize systematically).

3.3 Variable rate code

The author has devised a new simple bi-level code with easily controllable (d, k) characteristics and which can be made to have near-optimal efficiency. The code is amenable to writing in parallel over multiple tracks and has a variable bit rate: its rate is dependent on the data written. Experimental work carried out shows that this code is practical and the single-track form is highly suitable for Bayesian-style detectors.

Most popular codes make an implicit assumption that the bit rate must be constant. In a disk or pre-formatted tape system this is essential for a constant sector size, but in some applications there is no such requirement.

Digital information compression is now commonplace, and can make the data stream appear almost random. This increases the entropy characteristics of the data in a way similar to that of a scrambler. This means that there is no requirement to break up long runs of the same symbol which may confuse the clock recovery system. The variable rate codes presented will, however, benefit modestly by simple symbol counting and representing the most commonly found symbols with the shortest symbols—a small symbol map at the beginning of a data run can substantially improve the bit density with minimal overhead.

3.3.1 Description of the code

This code was envisaged for use in a bi-level environment, such as a magnetic recording channel. It represents a symbol by the time between transitions in the channel. However, unlike the variable rate codes' closest relative, BPPM codes [94], there is no information stored in the phase of the transition and the bit rate is variable.*

Single track operation

S_i = *symbol representing* $\log_2 b$ *bits*

First, the data stream is split into symbols (S_i). That is, the stream is represented by a very large number written in some base, b, with each digit corresponding to a symbol. Bases formed from binary ($b = 2^n$) are useful. Each S_i has a value from 0 to $b - 1$.

t_i = *inter-transition interval*

The value of the symbol then determines the time, t_i, between transitions. The interval t_i is granular (i.e., it only takes on certain values), and is referred

T = *clock period*

to some stable clock of period T.

A simple mapping of symbol to interval is

$$t_i = (1 + d + S_i)T \tag{3.1}$$

*In fact, in the $(d, k) = (1, 3)$ code presented later, the *first* transition defines a clock reference, so the sign of the phase does represent one bit over the entire transmission. Other arrangements can be trivially made to eliminate this bit though.

where d is the minimum interval from the chosen (d, k) values. In addition, k can be calculated (or chosen) easily.

$$k = b + d - 1 \tag{3.2}$$

Hence the code and number of symbols can be derived directly from the (d, k) requirements.

For example, a simple two symbol code can be formed with $d = 0$, $S_i \in \{0, 1\}$. This code therefore gives a long state for a 1, and a short state for a 0. (The word 'state' refers here to the instantaneous value of the coded signal, either a 'high' or 'low' in a two level channel.) A similar code is used in some hand-held infra-red remote controllers.

Multiple track operation

Suppose there is a stream of symbols coming in, and the code must be written out. Given n channels (tracks) the input stream can be multiplexed over them. An example algorithm is

1. find a 'free' track, waiting if necessary
2. start writing the single-track symbol
3. mark the track not free until the symbol has finished
4. immediately go back to the first step

It is assumed here that the algorithm for finding a free track is deterministic (given a certain state of tracks, it will always find a particular track free) and that all tracks are running from the same clock. It must also be assumed that the processor can process all the free tracks in much less than one clock period. An example trivial algorithm for determining a free track is

1. start at track $u = 0$
2. if track u is free, use u

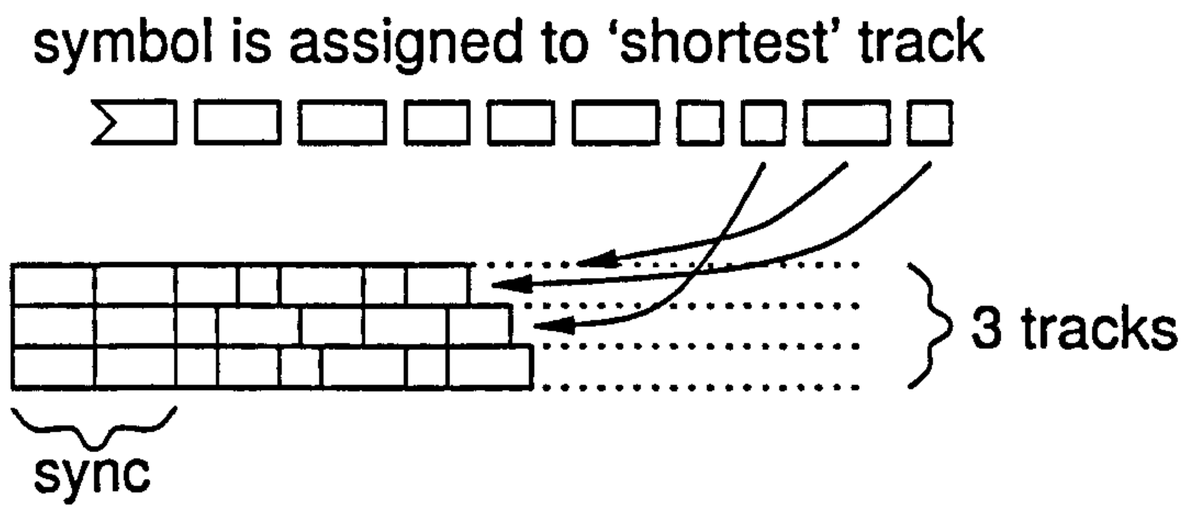


Figure 3.2: Expanding variable rate code to multiple tracks.

3. $u \longleftarrow u + 1$

4. if $u = n$ then no tracks are free, so wait a clock period

5. go to step 2

Now the modulation process will write a unique pattern over all the tracks for a particular input pattern. This algorithm does not attempt to be as efficient as the two-dimensional codes [24], but it does remain reasonably resistant to track skew.

Figure 3.2 shows the basic principle: each symbol is assigned deterministically to the first track that can accommodate it.

A practical code, $(d, k) = (1, 3)$

Take $(d, k) = (1, 3)$ giving an interval of $2T$, $3T$ or $4T$. Symbols will be manipulated in pairs, $[S_{2i}, S_{2i+1}]$, called quasi-symbols, Q_i. Each symbol can take on one of three values, $S_i \in \{0, 1, 2\}$, hence each quasi-symbol can take on one of 9 states, which are partitioned into 8 data states ($0 \leq Q_i \leq 7$) and 1 sync state ($Q_i = 8$). Since the sync state is not often used, it will be given the longest quasi-symbol state, $[2, 2]$.

The input stream is split into quasi-symbols of 3 bits ($b = 8$) and these are mapped into a data state Q_i which is split into two (normal) symbols (S_{2i}, S_{2i+1}) which are written to the tracks S_{2i} then S_{2i+1}.

Given a maxentropic (random) stream of input data, the expected density ratio, R, can be calculated [66].

$$R = \frac{\text{data density}}{\text{transition density}} \tag{3.3}$$

$$= \frac{x}{y} \cdot (d+1) \tag{3.4}$$

$$= \frac{3}{5.75} \cdot 2 = 1.04 \tag{3.5}$$

where

x = number of input data bits per quasi-symbol

y = clock periods the input bits are mapped into

$$\approx 2^{-x} \sum_{Q_i=0}^{7} (d+1+S_{2i}+d+1+S_{2i+1}) \tag{3.6}$$

$$= \frac{1}{8} \cdot [(0+0)+(1+0)+(0+1)+ (2+0)+(0+2)+(1+1)+ (2+1)+(1+2)+32] \tag{3.7}$$

$$= \frac{46}{8} = 5.75 \text{ clocks (on average)} \tag{3.8}$$

In practice, R should be slightly better because the most common values are likely to be assigned to the shortest symbols.

The efficiency of this code is close to 98% (this figure is derived in appendix B section B.1.7). The redundancy is introduced by the sync symbol, which cannot be used to code data, but can be very practical when decoding data.

Figure 3.3 shows all the possible quasi-symbols and figure 3.4 shows how these are strung together to form a coded data stream.

Decoding

Decoding is not difficult in single track mode. In multi-track mode the tracks should be kept clock aligned as far as possible and if the track skew becomes greater than the clock period, then the demodulator should compensate for this. (The k is low enough in the example $(d,k) = (1,3)$ code that, given a suitable sync sequence over all tracks at the start of the data run, the track skew can in

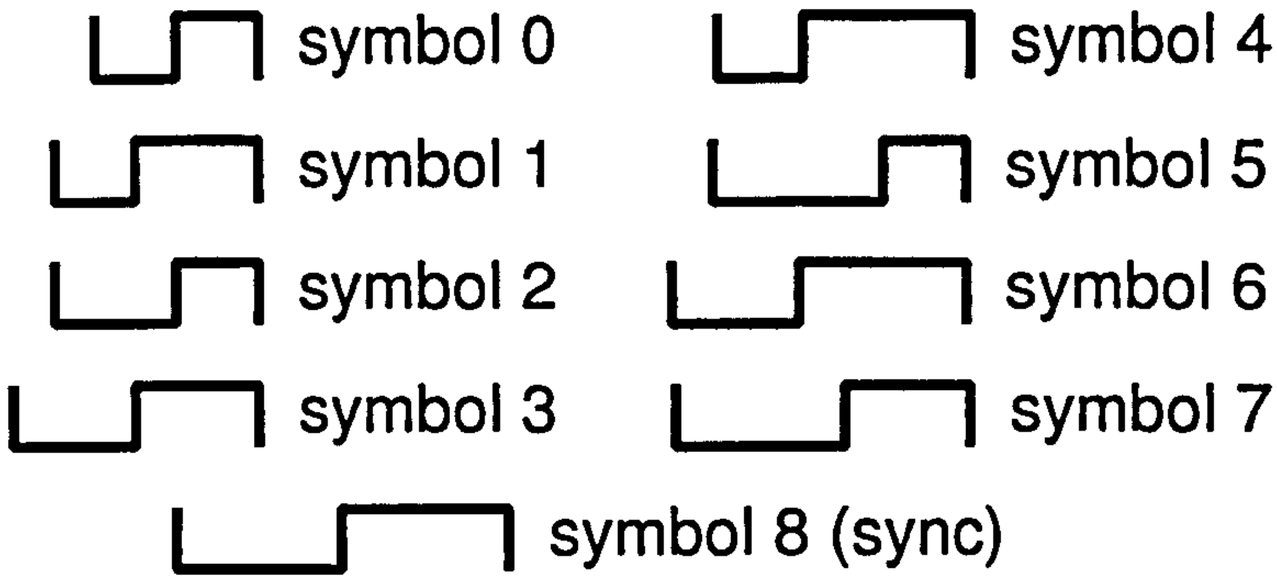


Figure 3.3: The possible quasi-symbols.

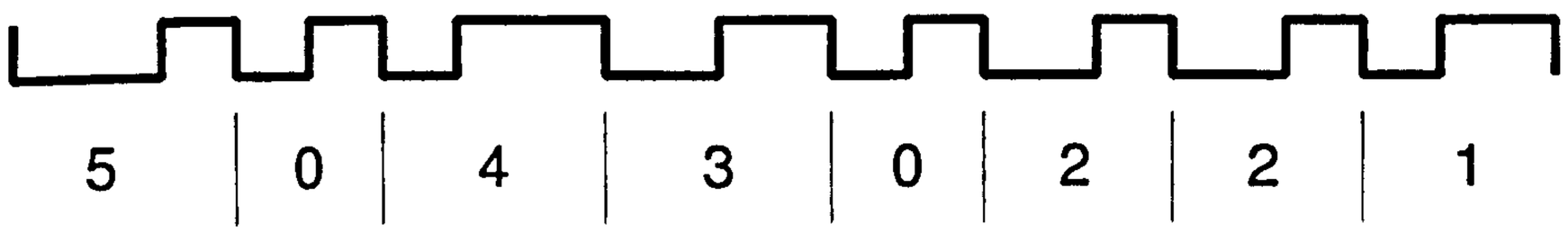


Figure 3.4: Stringing together quasi-symbols from a data stream to make a coded stream: each number represents 3 bits from the input data.

principle be followed: each track can be treated as an isolated single-track and demultiplexed later.) The decoder simply times the inter-transition gap on each track. Since the decoder knows the free track algorithm, if two or more tracks change at once the written order can be unambiguously determined. Then the decoder boxes up the symbols into quasi-symbols (and sync) to be processed.

A single read error in multi-track mode can propagate though the code by upsetting the track skew correction mechanism, although it should be possible to guard against this in single track mode: a single symbol error only affects a single quasi-symbol (or two in the case that it is between quasi-symbols). Each quasi-symbol starts on a high to low transition, so quasi-symbol resynchronization is a trivial matter.

Because each inter-transition period only represents one symbol, the channel characteristics can also be processed in a simple manner if the ISI is less than d. A particular symbol will give a characteristic inter-transition period which may only loosely be related to the t_i that generated it. In this case, there should be

ISI: inter-symbol interference

3 definite inter-transition periods, mapping to $t_i = 2$, $t_i = 3$ and $t_i = 4$ in the example given.

3.3.2 Comments

If the system permits it, a form of write precompensation can be easily applied. By making $t_i = s_0 + g(S_i)$. A simple example might be $g(0) = -0.2$, $g(1) = 0.9$, $g(2) = 2.0$. However, this write precompensation could give problems with the multi-track skew compensation if g can take non-integer values. Another example for use with large b is $g(j) = x^j$ where x is constant (e.g., 1.2), which gives interesting jitter characteristics, although it has not been tested in the credit card application.

3.4 Code summary

NRZ: non-return to zero

Table 3.2 lists the capacity of raw (d, k) codes relative to NRZ: $(d, k) = (0, \infty)$. (It was derived using equation (B.12) in appendix B.) Table 3.3 lists some common (or interesting) code modulation schemes, their (d, k) values, rate, RDS, timing window, efficiency (the code's capacity relative to its (d, k) capacity) and density ratio (a measure of how many flux changes are used to code a bit). The definition of density ratio is given as equation (3.3). Appendix B shows how to derive efficiency and capacity. Also included in the table is the new measurement, the TER-100, described in section 3.2. This measures the code's ability to recover from timing errors or large data errors.

RDS: running digital sum

3.5 What makes a 'good' code?

The best code for an application is not necessarily the one with the highest density ratio. Although the density ratio is a useful measure of probable bit density, other factors also play a part; especially in an environment as harsh as a credit card reader.

	k								
d	1	2	3	4	5	6	7	8	∞
0	0.69	0.88	0.95	0.98	0.99	0.99	1.00	1.00	1.00
1	.	0.41	0.55	0.62	0.65	0.67	0.68	0.69	0.69
2	.	.	0.29	0.41	0.46	0.50	0.52	0.53	0.55
3	.	.	.	0.22	0.32	0.37	0.41	0.43	0.46
4	.	.	.	.	0.18	0.27	0.31	0.34	0.41

Table 3.2: Capacities of (d, k) codes.

code	(d,k)	code rate	RDS %	timing wnd. $/T$	eff. %	density ratio	TER-100
NRZ (binary)	$(0,\infty)$	1.00	100	1.00	100	1.00	1.2±5.4
FM	$(0,1)$	0.50	0	0.50	72	0.50	1.4±3.4
MFM [67]	$(1,3)$	0.50	33	0.50	91	1.00	4.2±7.5
ZM [77]	$(1,3)$	0.50	0	0.50	91	1.00	46
RLL $(1,7)$ [48]	$(1,7)$	0.67	33	0.67	98	1.33	81
RLL $(2,7)$ [32]	$(2,7)$	0.50	33	0.50	97	1.50	68
3PM [47]	$(2,11)$	0.50	50	0.50	92	1.50	66
BPPM-$1_{S=2}$ [94]	$(1,5)$	0.40	20	0.50	62	0.80	5.5±14
BPPM-$1_{S=4}$ [94]	$(3,13)$	0.27	27	0.25	60	1.09	5.9±5.7
BPPM-$2_{S=4}$ [94]	$(1,6)$	0.43	43	0.50	64	0.86	15.7±15
variable rate codes (section 3.3)							
$(1,3)$	$(1,3)$	0.52*	33	0.52*	98	1.03*	2.5±3.6
$(1,4)$	$(1,4)$	0.57*	43	0.57*	100	1.14*	2.1±3.8

* average values

Table 3.3: Statistics for common modulation codes. Several values were taken from [23, 66]. The rate is the (average) number of clock periods used to encode 1 bit of data. The RDS (running digital sum) is a measure of magnetic disparity: the worst-case ratio of magnetizations in one direction against the other. The timing window is measured relative to the bit-clock $(1/T)$, and measures the required stability of the sampling rate. The efficiency (eff.) is the number of bits encoded relative to the maximum number of bits that could be encoded by a perfect (d, k) code. The density ratio is a worst-case measure of the number of effective bits encoded per magnetic flux change (assuming a fixed rate code). The TER-100 (timing error resilience) is an average measure of the number of bit errors after a timing error event.

The most important factor in this credit card application is timing jitter: the amount of error in determining the position of magnetic transitions in the medium. The density ratio can be made arbitrarily high if the clock is perfectly stable. However, when k is constrained by jitter (and hence d is also constrained) the density ratio is constrained too. For example, the clock recovery mechanism must place a transition to within 5% of its proper time with a $k = 10$ code. With a $k = 3$ code this is relaxed to nearly 20%. On a credit card the impulse response can vary, as well as the proper clock speed. This places a very severe restriction on k: FM code (the standard) has $k = 1$ while $k = 3$ is the approximate limit before errors when reading the magnetic stripe become likely.

Efficiency is important, but nearly all codes are close to full efficiency anyway, so there is very little to choose between the codes on this basis.

RDS: running digital sum

The RDS is not particularly important in the credit card channel. It could be argued that having a net magnetic moment on a credit card may have some detrimental side-effects (the long range magnetostatic fields may have a demagnetizing effect on other cards). However, this has not been an observed problem in other magnetic media, but its effects have not been studied. Possibly more problematic is the way that DC magnetic levels can shift the base level magnetization. This was an enormous problem for fixed threshold peak detectors in the past, but a slight asymmetry will not affect modern detectors to any great degree. Indeed, MR heads themselves introduce both DC offsets and non-linearity features into the read-back signal.

MR: magneto-resistance

A feature not often examined in coding theory is error recovery. Some codes are particularly bad at recovering from errors. There are two types of failure that appear in decoders: timing errors (where there is a discrepancy between the write clock and the read clock) and value errors (where a bit or group of bits are in error). It is usual to choose a (d, k) level that causes negligible timing errors, and make value errors dominant (but still low). If this is the case, then it is also useful to keep the timing recovery clock stable enough to flywheel over channel burst errors too.

Chapter 4

Clock recovery

Clock recovery is the regeneration of the time base of the originally written signal from the read-back signal. Timing recovery is as important as signal detection (chapter 5), if not more so because as yet there are no error correction schemes for timing faults. It has not, however, been extensively researched in industry* because in most applications the magnetic medium velocity can be carefully controlled: in a hard disk the speed change is only up to ~0.1% over a few million bits. Helical scan tape recorders also have fixed velocities derived from a crystal oscillator, and normally only have to contend with phase matching.

In the near future there is likely to be more emphasis placed on timing schemes, as systems become less tolerant to timing changes (smaller bit lengths) and physically smaller (and so have less inertia and are more vulnerable to environmental and self-generated vibration).

This chapter describes the conventional clock recovery algorithms and then develops a new phase and velocity estimation scheme based on snatches of nearly alike signals.

---

*At least, not to the extent that coding and detection have been studied.

4.1 Standard clock recovery mechanisms

The traditional, and successful, method of timing recovery is to use a small piece of training code at the beginning of a data sector to align a phase locked loop (PLL). The PLL can then adjust itself slightly from any self-clocking properties of the code in the data sector. This form of slight 'tweaking' of clock speed is not accurate enough for high density recording on a credit card, where speed changes may be more than 200% during a few thousand bits. Indeed, simpler schemes can be are more effective.

PLL: phase locked loop

4.1.1 Timing window

The timing window is the maximum interval of discrepancy between the read-back signal and the decoder's clock. For instance, the RS-232 serial data transmission standard specifies transmission with 8-bit bytes framed with a start bit and followed by eight binary data bits. The receiver finds the centre of the start bit, and triggers its own clock. It can then sample each following data bit at intervals of the bit rate. Barring noise, if the receiver's clock speed is within 6.25% (a timing offset of ± 0.5 of a bit over 8 bits) of the transmitter's clock speed, the bits will be correctly received. The receiver can resynchronize itself every byte. With a clock derived from a crystal oscillator (and square edges) this is a very easy proposition.

In general, the timing window of a channel is dependent on the code used. In the NRZ case, half a bit off-centre in either direction will cause a mis-read. This gives NRZ a timing window of 1. For most codes, the timing window is equal to the rate of the code: any mistake in the *code* clock will result in a mistake in the *data* decoded. Some sparse codes can have looser timing windows.

NRZ: non-return to zero

4.1.2 Low density variable speed systems

Low density, as a convenient definition, is a density at which ISI is not noticeable. If it is possible to distinguish individual peaks or zero-crossings, then it is

ISI: inter-symbol interference

possible to time the interval between them.

On the standard credit card, FM code is used. This is a $(d, k) = (0, 1)$ code, hence the interval between each peak is either $1T$ or $2T$ where T is the code clock interval. At low densities the clock can be recovered on a bit-by-bit basis by comparing the most recently read interval against $1.5T$. If the interval is $< 1.5T$, i.e., approximately $1T$, the decoder must wait for the next interval (which will also be about $1T$) and a `1` is output as the bit value. A new value of $1.5T$ can now be calculated from the sum of the two intervals. If, on the other hand, the most recent interval was $> 1.5T$, i.e., approximately $2T$, a value of `0` is output and $1.5T$ updated from the single interval. This method is extremely robust, and can cope with up to 50% speed change per bit. It can fail if there is a long run of `1`s, but will immediately reset itself once a `0` is detected.

Another code, used for recording data onto magnetic camera film strips, is pulse position modulation (PPM) [96]. In this scheme the clock is again recovered on a bit-by-bit basis. It is not easily classified by normal code measures, because it was designed specifically to cope with low density recording at variable speed. Each bit comprises three magnetic transitions: the outer transitions are shared with neighbouring bit cells, giving a net density of two transitions per bit. The outer transitions mark the boundary of the bit cell. The value of the bit now depends on whether the central transition is in the first half of the cell or the second half. This method allows up to 100% speed change per bit.

PPM: pulse position modulation

Both these methods rely on low density properties: that is to say, the transitions can be located accurately in the code stream.

4.1.3 External clock

Multi-track linear tape systems, like DCC [79], often rely on a separate, auxiliary, clock track to provide timing cues. This provides advantages in terms of freedom of tape velocity and freedom from self-clocking codes at the expense of a track that could have been used for data. This use of a whole track is normally wasteful compared with a self-clocking system or a two-dimensional

DCC: digital compact cassette

self-clocking code [24]. However it reduces the complexity of the decoding electronics and the clock can also utilize an area the tape that may otherwise not be suitable for data, as only the clock's phase is required so it is not susceptible to short drop-out losses.

Both external clocks and two-dimensional codes can suffer from the effects of tape skew: the phenomenon of the tape becoming warped or of the azimuth angle between the head and the tape being non-zero [26]. Tape skew is a dynamic feature, and additional compensation, possibly using the self-clocking properties of the code, is required if the bit length is less than the characteristic skew size.

4.1.4 Sensitive phase locking

In stable time-base systems, variations of the phase locked loop (PLL) and digital phase locked loop are used to update the frequency and phase of the sampling clock and these methods have proved reliable and durable [71]. The minimum mean squared error (MMSE) is often used as a suitable metric to find an optimum sampling point [81] and this can be tailored to the particular code [1]. Another method is based on the known impulse response of the channel which can lead to less PLL jitter [86].

MMSE: minimum mean squared error

The basic principle of the MMSE method is that the error between the desired signal and the received signal is a minimum when the sampling clock (the clock in the read-back mechanism) is synchronized with the data clock (the clock that the medium was originally written with, that determined the bit length).

Consider an error signal, e_k, made from the difference between the ideally sampled signal, $\hat{r}_k$, and the actual sampled signal, $r_k(\Delta) \equiv r(kT+\Delta)$. The latter signal is a function of the timing offset, Δ.

e_k = *error*
k = *sample index*
$\hat{r}_k$ = *ideally sampled signal*
r_k = *sampled signal*
Δ = *timing offset*

$$e_k(\Delta) = \hat{r}_k - r_k(\Delta) \tag{4.1}$$

It should be noted that $\hat{r}_k$ can usually be generated from, say, the convolution of $\hat{w}_k$, the decoded channel data, and the channel response.

To form the MMSE condition, $e_k^2(\Delta)$ should be minimized with respect to Δ.

$$\frac{d(e_k^2)}{d\Delta}(\Delta) = 2e_k(\Delta)\frac{de_k}{d\Delta}(\Delta) \tag{4.2}$$

$$= -2\left(\hat{r}_k - r_k(\Delta)\right)\frac{dr_k}{d\Delta}(\Delta) \tag{4.3}$$

$$= -2\left(\hat{r}_k - r_k(\Delta)\right)\frac{dr}{dt}(kT+\Delta) \tag{4.4}$$

Assuming that the Nyquist sampling constraints are satisfied, the gradient can be approximated as

$$\frac{dr}{dt}(kT+\Delta) \approx \frac{1}{2T}\left(r_{k+1}(\Delta) - r_{k-1}(\Delta)\right) \tag{4.5}$$

LMS: least mean squares

and hence, using a gradient descent method similar to the argument for the LMS inverse filtering algorithm (the explanation of which is postponed until section 5.2.2, page 110, where it is examined more fully), the sampling clock's phase can be updated as follows.

$$\Delta \longleftarrow \Delta + \mu\left(\hat{r}_k - r_k(\Delta)\right)\cdot\left(r_{k+1}(\Delta) - r_{k-1}(\Delta)\right) \tag{4.6}$$

As with the LMS equation case, μ is used as an adjustable step size that is chosen with regard to the timing characteristics. (The choice of this step size can be quite critical, especially in adaptive systems [75].) Eventually Δ will be optimised with respect to the mean squared error and this is (hopefully) also the point where the true timing offset is zero.

There are many simplifications used to speed up the calculation, one common method works by quantizing of the error signal

$$\Delta \longleftarrow \Delta + \mu\,\mathrm{sgn}\left(\hat{r}_k - r_k(\Delta)\right)\cdot\left(r_{k+1}(\Delta) - r_{k-1}(\Delta)\right) \tag{4.7}$$

which can be implemented very easily in hardware. In the above, sgn is the

signum function:

$$\operatorname{sgn} x = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases} \tag{4.8}$$

These standard algorithms all rely on the clock being stable to operate successfully. Any frequency adjustment is normally performed by continually adjusting the phase and feeding the heavily averaged phase adjustment back to the frequency generator (hence closing the loop in a PLL).

4.2 Snatches of signal

A 'snatch' is defined here as a small piece of sampled signal. A snatch is often already present in detectors in the sampled analogue data buffers. The standard clock recovery mechanisms described previously can work with single sample values from the data stream; what follows is a method of clock recovery using many samples. The advantage of this new method, developed by the author, is that it not only allows clock phase corrections, but also direct clock *frequency* corrections. This is necessary for the rapidly changing velocities encountered in credit card detection. Another novelty is that the method has no long-term memory: each result is based only on the data presented at that time. (However, only a velocity *offset* is produced, so when decoding a stream there is memory in the absolute velocity.)

This rest of this section presents a method of calculating timing and velocity offsets from snatches; starting with identical pairs of signals and working through to see the effects with non-identical signals. In a system, one snatch is read from the channel itself, and the other is derived from the decoded channel output.

4.2.1 Snatch offset estimation

If there are two identical snatches, it is fairly easy to calculate any slight timing offset between them. The correlation function is appropriate here. Another, less

direct but computationally simpler, method is now explained. The principle is to form Taylor's expansion of the difference between the two snatches. This method requires that the offset is fairly small (smaller than the characteristic wavelength of the code) otherwise incorrect phase adjustments will result.

Consider the snatches of signal shown in figure 4.1, $r(t)$ and $r(t+\Delta)$. Taylor's expansion is

r = read-back signal
Δ = small time offset

$$r(t+\Delta) = r(t) + \frac{\Delta}{1!} \cdot \frac{dr}{dt}(t) + \frac{\Delta^2}{2!} \cdot \frac{d^2r}{dt^2}(t) + \cdots. \tag{4.9}$$

Just taking the first-order expansion (i.e., Δ small) gives

$$r(t+\Delta) - r(t) \approx \Delta \frac{dr}{dt}(t). \tag{4.10}$$

All the terms in this equation, with the exception of the value of Δ itself, are known. Perhaps to find Δ from a slightly noisy source, the following could be tried:

$$\int_0^\tau r(t+\Delta) - r(t)\, dt \approx \Delta \int_0^\tau \frac{dr}{dt}(t)\, dt. \tag{4.11}$$

The integration is simply an averaging of the differences in the signals. The gradient of the signal, $\frac{dr}{dt}(t)$, is likely to be noisy, so integrating over the length of the snatch, τ, will remove some of this. Rewriting this equation to reflect the fact that the integrals are a constant property of this particular snatch, equation (4.11) reveals

$$Z_n \approx \Delta \cdot Y_n \tag{4.12}$$

where Z_n and Y_n are the computed values of the integrals. Now Δ can be found trivially.

However, equation (4.11) is *not* a good way of finding Δ. A typical (d, k) or magnitude constrained signal will average about zero, making Z_n and Y_n nearly zero and hence Δ very noisy. This is easily explained because the simplification of the right hand side of (4.11) is simply (the noisy subtraction) $r(\tau) - r(0)$.

Consider instead the following equation, which tries to combat this prob-

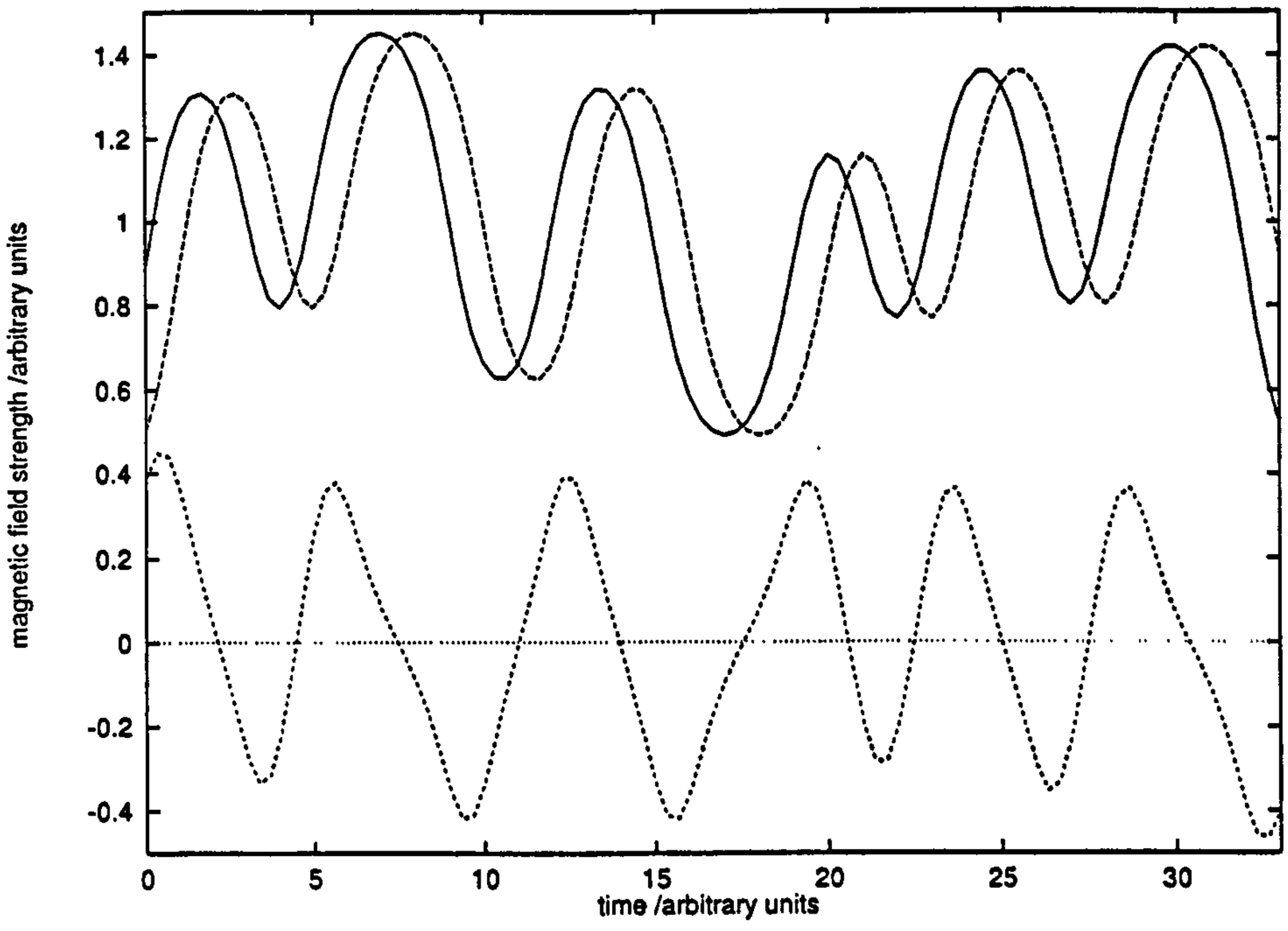


Figure 4.1: Modeled magnetic signals: one signal is delayed by 1 time unit. The upper half shows $r(t)$ and $r(t+\Delta)$. The lower half shows $r(t+\Delta) - r(t)$.

lem. This is simply (4.11) with an added $\frac{dr}{dt}(t)$ term.

$$\int_0^\tau \Big(r(t+\Delta) - r(t)\Big)\,\frac{dr}{dt}(t)\,dt \approx \Delta \int_0^\tau \left(\frac{dr}{dt}(t)\right)^2 dt \tag{4.13}$$

$$\Longrightarrow \quad Z \approx \Delta \cdot Y \tag{4.14}$$

Now Y is always positive and substantial (i.e., not nearly zero). Z is also mostly positive (or mostly negative, depending on the signal offset), giving a far less noisy value of Δ. In fact, the difference between the signal and retarded signal is directly related to the usual definition of the gradient operator:

$$\lim_{\delta \to 0} \frac{dr}{dt}(t) = \frac{1}{2\delta}\Big(r(t+\delta) - r(t-\delta)\Big) \tag{4.15}$$

so equation (4.13) is really comparing *a* gradient calculated from the two signals against *the* gradient calculated from one signal. Another valuable feature

of (4.13) is that even when the offset *is* larger than the characteristic wavelength of the signal, the calculated Δ maintains the correct sign, although not the correct magnitude, for most signal patterns.

The conversion of (4.13) to the sampled domain is straightforward, but because Δ is not constrained to the sampling clock, $r(t+\Delta)$ shall be written as $\hat{r}_i$.

$$\sum_{i=0}^{\tau-1} (\hat{r}_i - r_i) \cdot \frac{1}{2}(r_{i+1} - r_{i-1}) \approx \Delta \sum_{i=0}^{\tau-1} (r_{i+1} - r_{i-1})^2 \tag{4.16}$$

It is assumed above that one sample interval is (several times) faster than the code clock interval, i.e., the signal is oversampled. Figure 4.2 shows some intermediate calculations on signals, and figure 4.3 shows the variation of Δ with offset for many synthesized code signal snatches, demonstrating the superior consistency of equation (4.13) against (4.11). The latter equation produces estimates that are variable and quickly divergent at large timing offsets whereas the former equation is more predictable and more likely to return the correct direction of the offset.

4.2.2 Snatch velocity offset estimation

Velocity difference is also measurable from two snatches, given that the two signals are in phase. Going back to Taylor's expansion, the following is evident given that δt is small.

δ = small velocity offset

$$r((1+\delta)t) - r(t) \approx \delta t \frac{dr}{dt}(t) \tag{4.17}$$

Thus for snatches, the following equation estimates the value of δ.

$$\int_0^\tau r((1+\delta)t) - r(t)\, dt \approx \delta \int_0^\tau t \frac{dr}{dt}(t)\, dt \tag{4.18}$$

The more stable version is again written

$$\int_0^\tau \left(r((1+\delta)t) - r(t)\right) \frac{dr}{dt}(t)\, dt \approx \delta \int_0^\tau t \left(\frac{dr}{dt}(t)\right)^2 dt. \tag{4.19}$$

$$\Longrightarrow \quad Z_v \approx \delta X_v \tag{4.20}$$

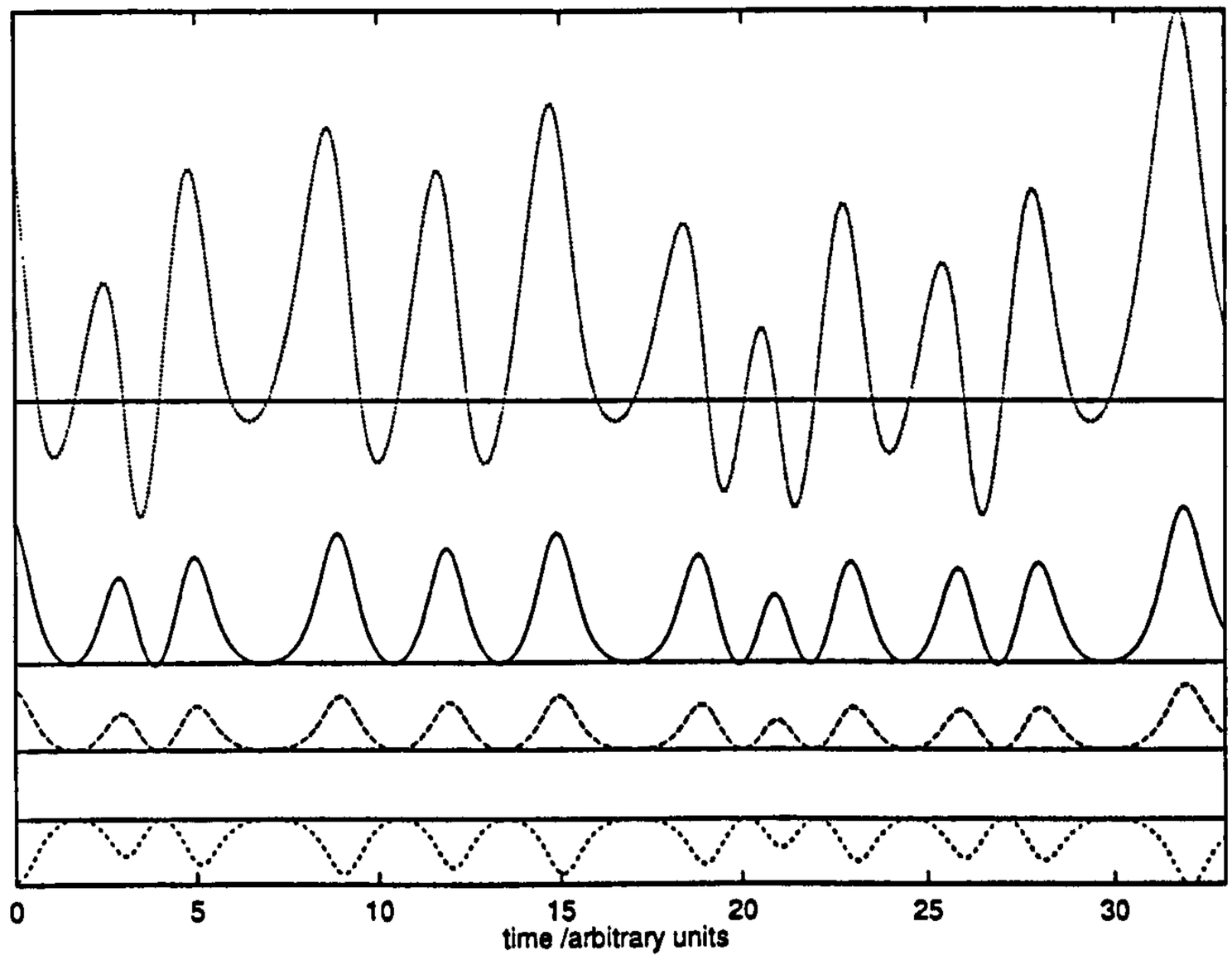


Figure 4.2: Intermediate signals used for offset calculation. All the traces show $(r(t+\Delta) - r(t))\,\frac{dr}{dt}(t)$ for the snatch shown in figure 4.1. Top to bottom: $\Delta = 2.0$, $\Delta = 0.5$, $\Delta = 0.2$, $\Delta = -0.2$.

Figure 4.4 shows signals with a speed mismatch and the difference between them. Figure 4.5 shows a range of the intermediate signals with velocity offsets and figure 4.6 shows the stability of the calculation in equation (4.18) with several modeled code snatches.

4.2.3 Snatches with non-identical signals

The system of velocity and offset calculations with snatches of signals was derived for identical signals. Although the system will work with one degraded signal, the results are not as precise. Consider, for example, the distortion function shown elsewhere in figure 5.14, page 116. The effect of the distortion to velocity offset estimation is a spread in δ, and a value of δ that is always slightly too small, see figure 4.7.

Using the threshold levels shown in figure 5.14 to 'flatten' both signals when

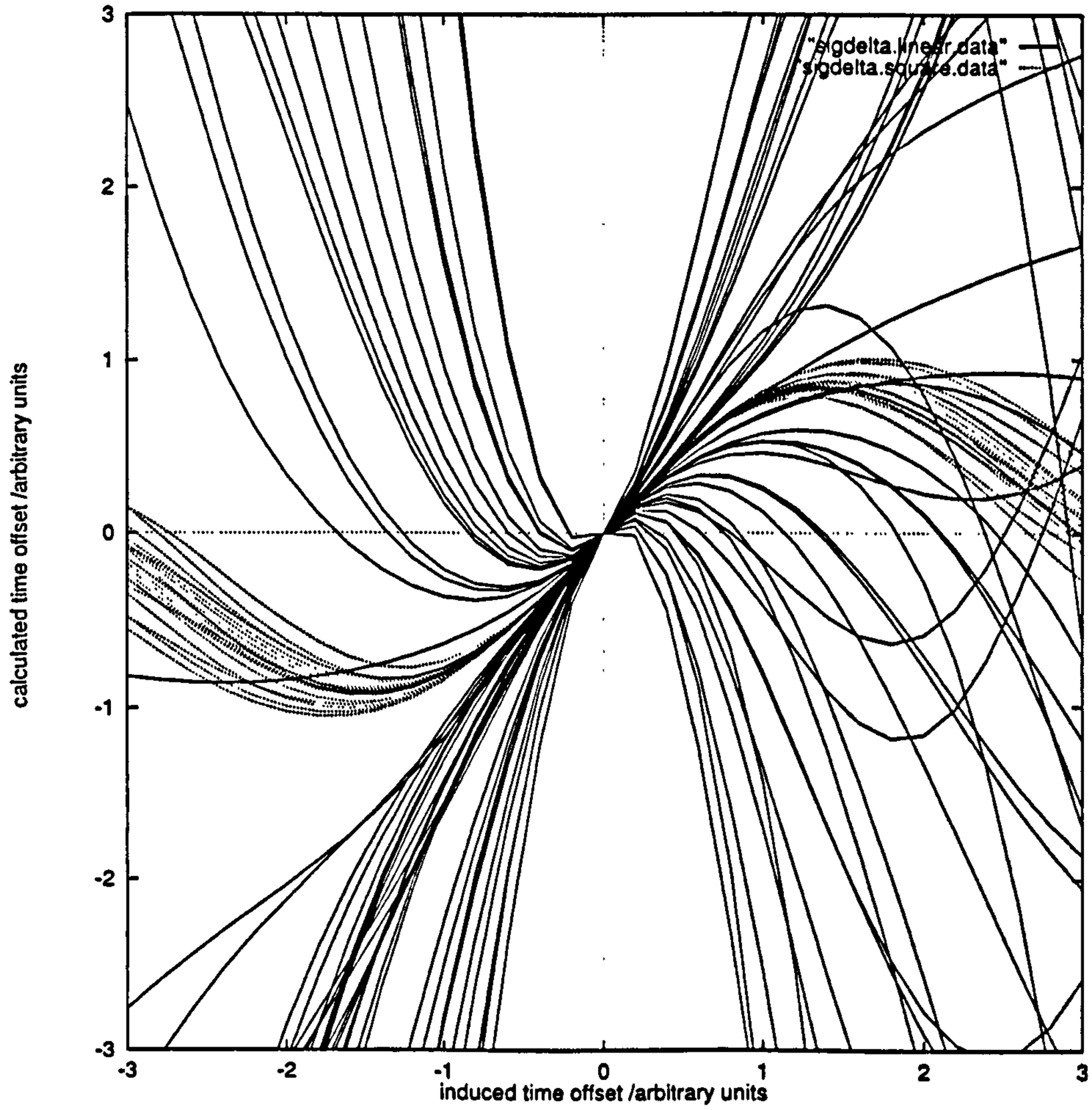


Figure 4.3: Plot of calculated Δ against induced time offset between many pairs of identical signals. Time is measured in code clocks, a $(d, k) = (1, 3)$ code was used. Solid lines were calculated using equation (4.11), dashed lines with (4.13). A perfect algorithm would give a single straight line, Δ=induced time offset.

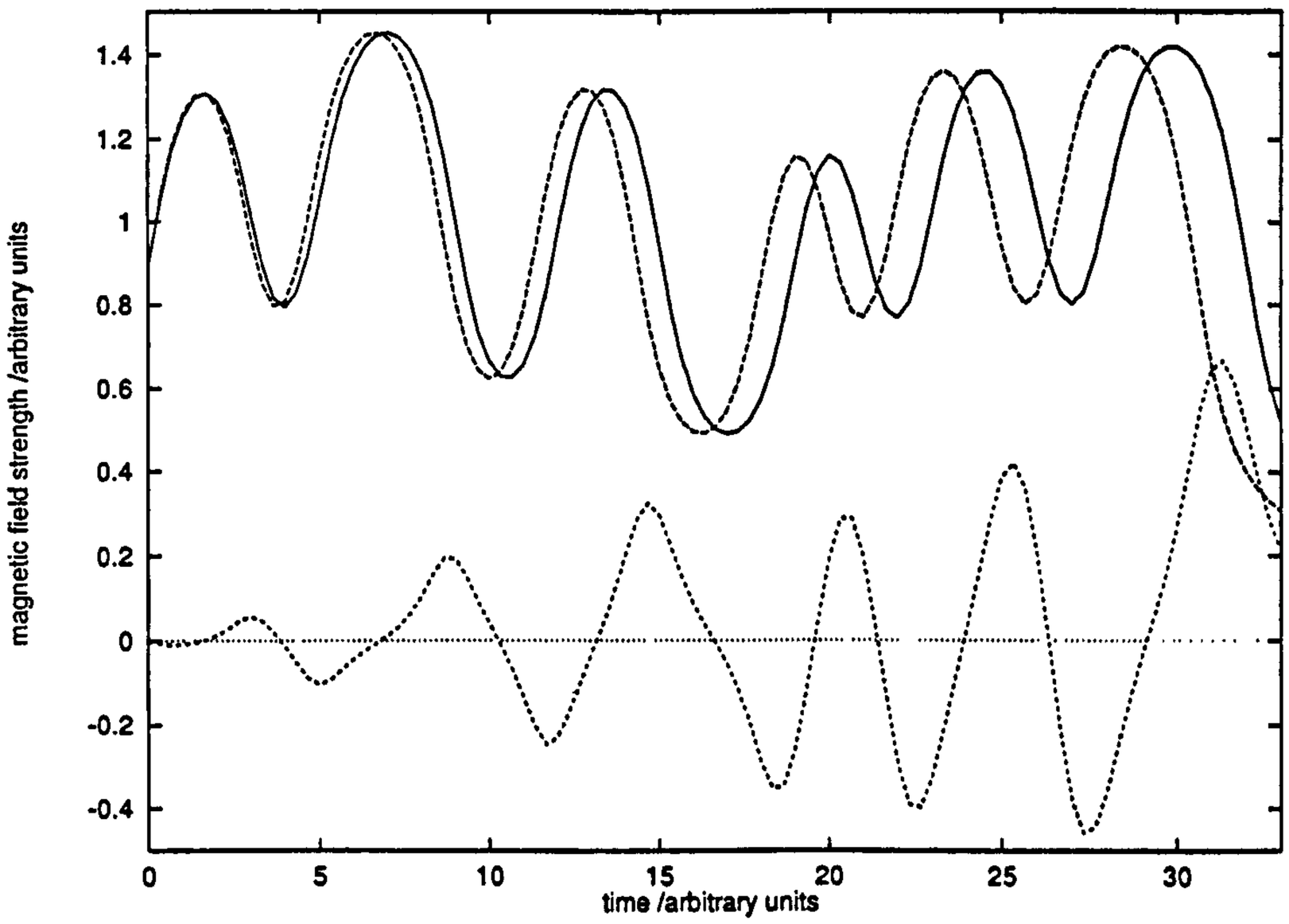


Figure 4.4: Modeled magnetic signals: one signal is advanced in velocity by 5%. The upper half shows $r(t)$ and $r((1+\delta)t)$. The lower half shows $r((1+\delta)t) - r(t)$.

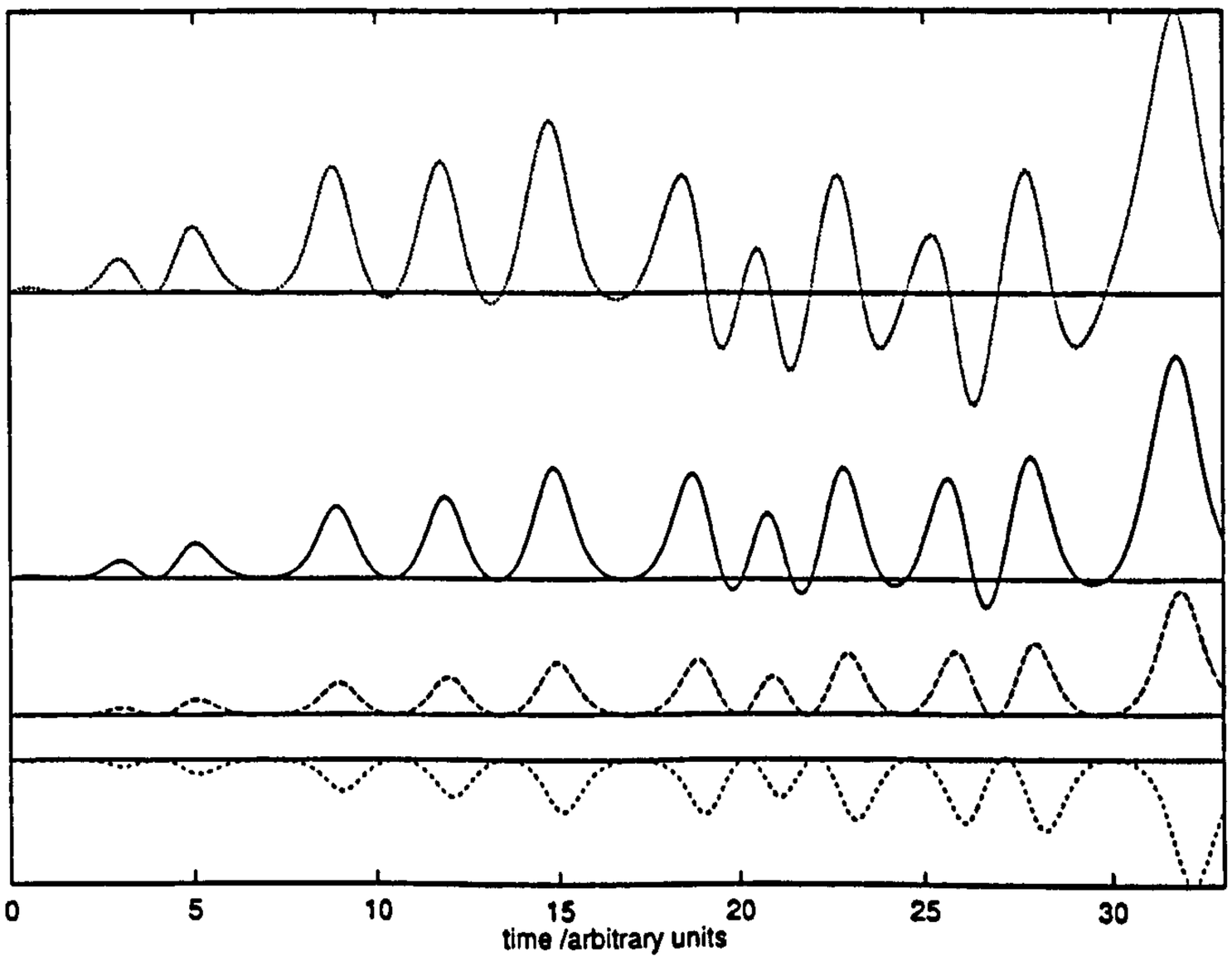


Figure 4.5: Intermediate signals used for velocity offset calculation. The traces show $(r((1+\delta)t) - r(t))\,\frac{dr}{dt}(t)$. Top to bottom: $\delta = 0.10$, $\delta = 0.05$, $\delta = 0.02$, $\delta = -0.02$.

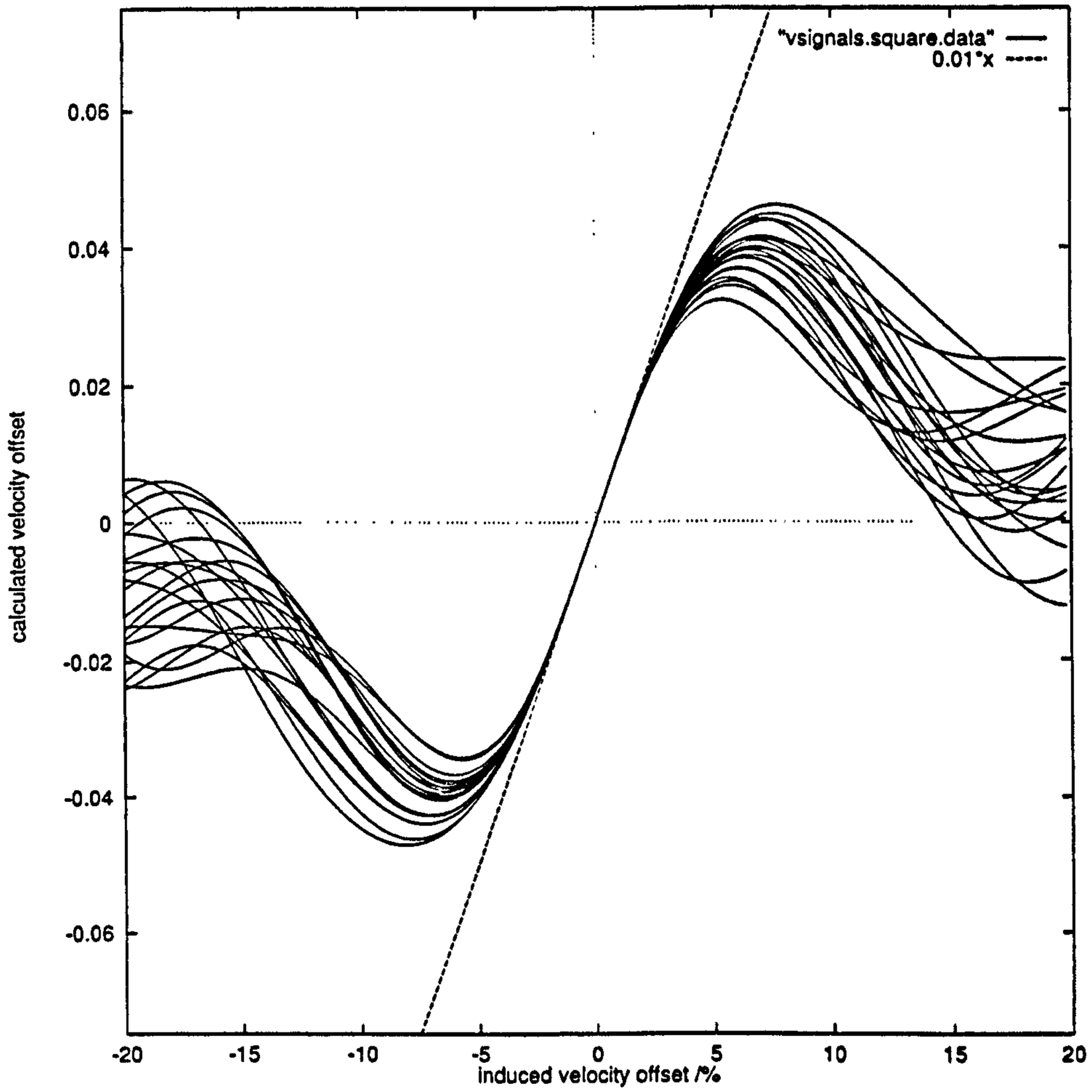


Figure 4.6: Plot of calculated δ against induced velocity offset between many pairs of identical signals. A $(d, k) = (1, 3)$ code was used over a period of 33 code clocks. The perfect line is also shown.

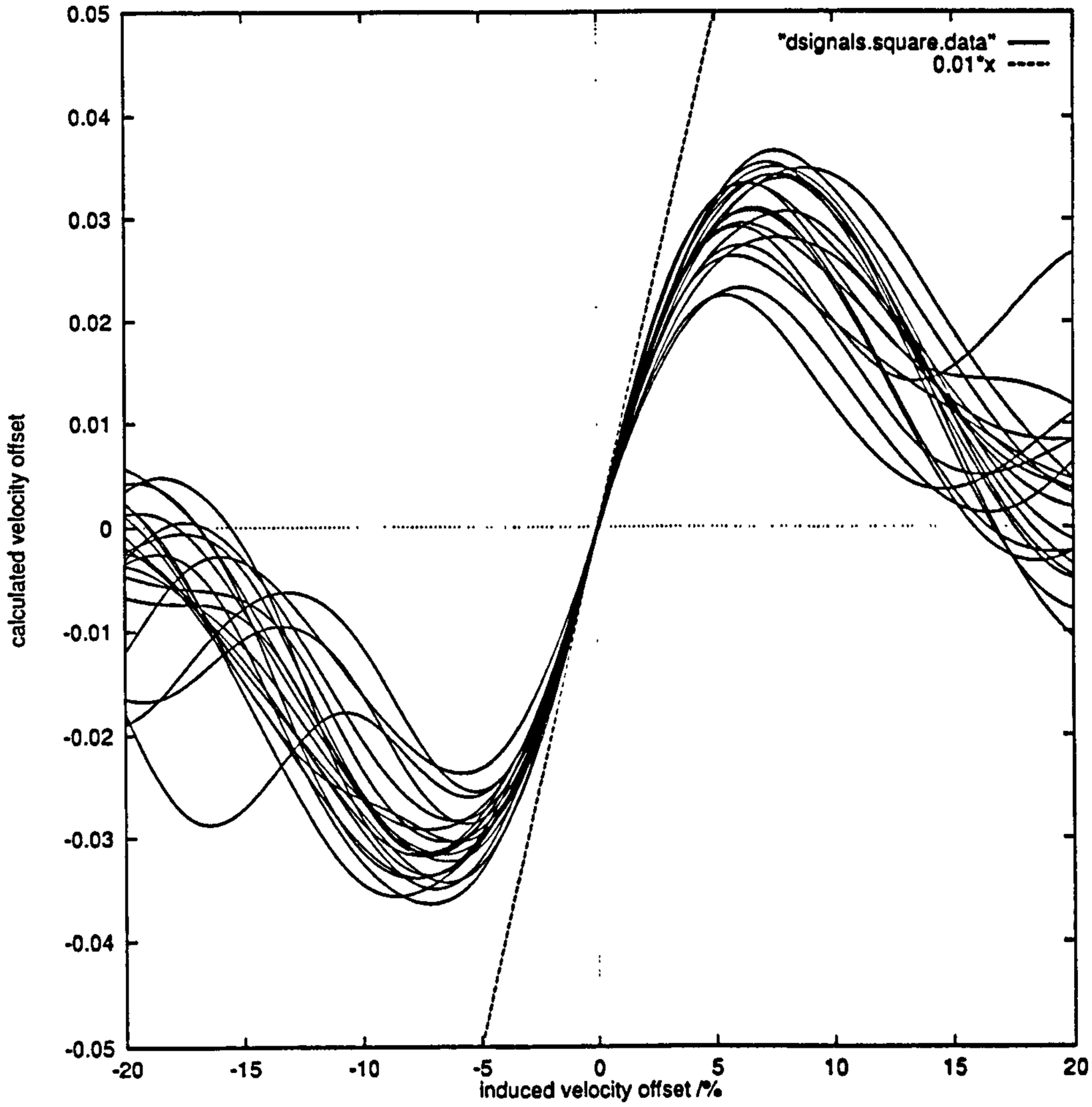


Figure 4.7: Plot of calculated δ against induced velocity offset between many pairs of signals, one of which has been distorted by an MR-like function. A $(d, k) = (1, 3)$ code was used over a period of 33 code clocks. The perfect line is also shown.

they are in the non-linear region has the, perhaps unexpected, effect of making the estimate worse, although marginally more consistent. The effect can be seen from figure 4.8. Consequently, when comparing signals there is nothing to be gained in this timing recovery system from trying to remove the non-linearity by truncation because it is better to be inconsistently correct than consistently incorrect. (However, one can envisage that it is possible to introduce a scaling factor to correct the shortfall in calculated velocity offset.)

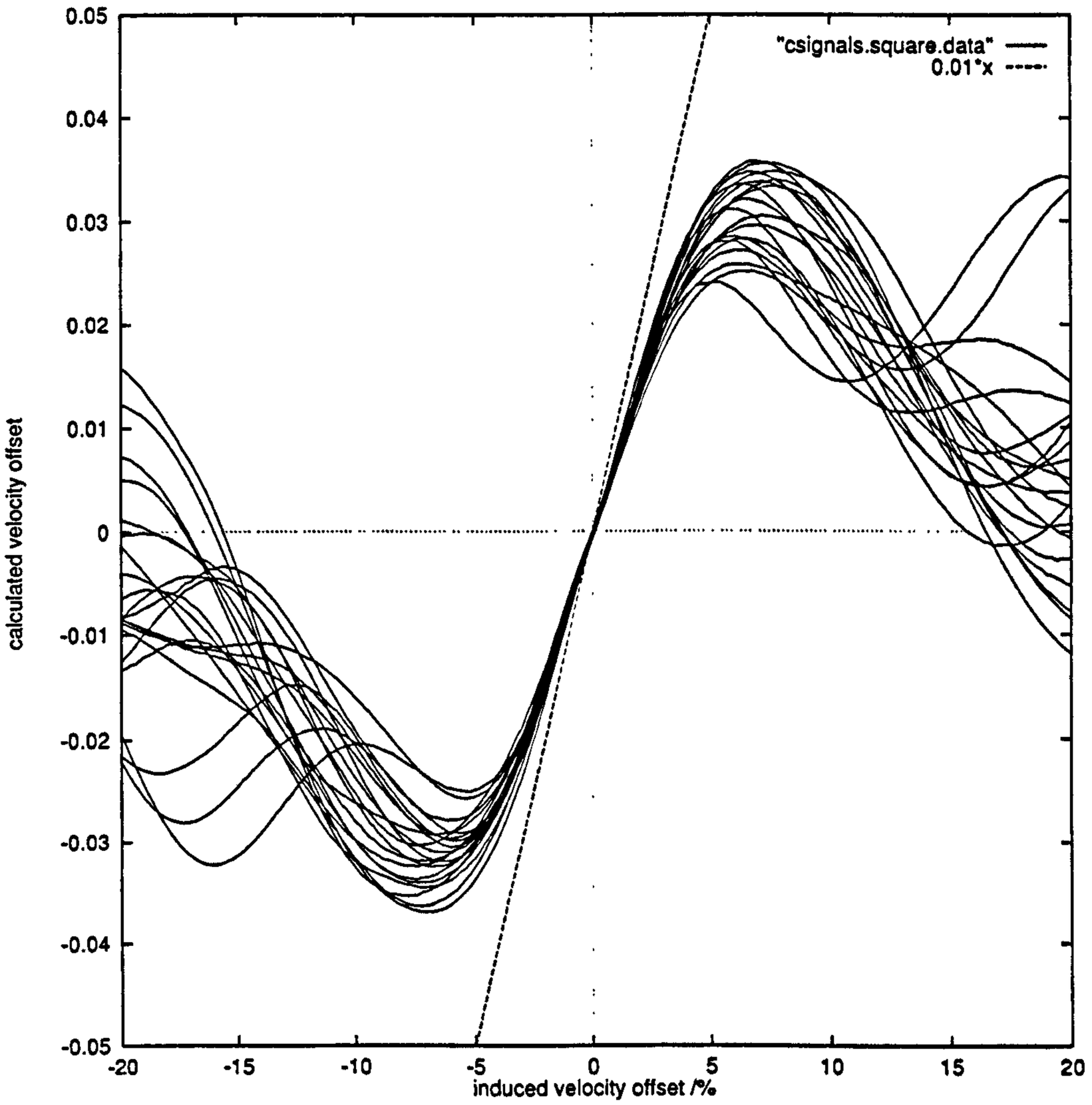


Figure 4.8: Plot of calculated δ against induced velocity offset between many pairs of signals, one of which has been distorted by an MR-like function. Both functions have been truncated when at the level where distortion becomes non-linear (see figure 5.14). A $(d, k) = (1, 3)$ code was used over a period of 33 code clocks. The perfect line is also shown.

4.3 What makes a 'good' clock recovery system?

The ideal clock recovery mechanism is fast, reliable, inexpensive and requires no overhead in the code itself. The last issue is moot with a $(d, k) = (1, 3)$ code as there is a great deal of redundancy already present—redundancy that is required to overcome the jitter in the signal, not deficiencies in the clock recovery mechanism.

The speed issue is related to the number of computation cycles it takes to update the phase. The standard PLL option uses one operation per cycle. The snatch method uses n operations for an n-sample buffer and a correlation uses n^2 operations. For small buffers this may not be a heavy penalty. However, the size of the buffers directly affects the amount of offset that the timing recovery mechanism can compensate reliably. With very large speed variations the PLL method soon starts to give erratic results because the timing is lost and errors in the decoded signal start to appear. Once an error appears the large step size, which is required for fast tracking, will throw out the PLL even more.

PLL: phase locked loop

There is a second reason, specific to changing channel parameters, that complicates matters further. The PLL algorithm requires that there is an estimate of the channel to compare against its reading: and estimating the channel to the required accuracy requires n operations in itself.

As hinted above, reliability—that is to say the noise margin and offset resilience in the clock recovered—is dependent on the snatch buffer size. A mechanism can only use the signal data that is currently stored in buffers with stored previous signal statistics. The smaller the buffers the larger the standard deviation of the present statistics.

The remaining issue, cost, is related both to the decoding time and circuit size and complexity. Both the snatch estimation and correlation methods are an order of magnitude more complex than the PLL circuitry. However, the importance of complexity cost is mitigated by the fact that there is not much data to decode and that it can be decoded off-line in software. This leaves the cost issue depending on the decoding speed required.

Given these criteria, the snatch velocity estimation method presents itself as an attractive new clock recovery mechanism for the credit card medium. The snatch *phase* estimation is not as useful because in this application the velocity changes too fast for the phase to compensate, and the velocity estimation can perform first-order velocity *and* phase correction. Nevertheless, in many other applications the computation required makes PLL methods the only valid choice. Whether this will still be the case with much smaller physical devices is not known at present. The snatch velocity estimation appears even more attractive when placed in the context of having a variable channel, where some of the results from the timing calculation can be used in updating the channel estimate, reducing the effective computation overhead. Appropriate behavior in the case of non-identical signals is also a positive point. However, this must be tempered with the realization that the algorithm is memoryless (apart from an initial velocity estimate which may have been derived from a previous calculation). In the case of very different signals (for example, during a signal dropout), it will produce very incorrect results.

Chapter 5

Equalization and detection

In electronics it is nearly always more difficult to go from the analogue to the digital domain than vice versa. In this chapter, the process of equalization (or linear equalization) is the analogue process of recovering the analogue signal that was written to the channel from the analogue read-back signal. The channel will have distorted this signal and the process is not necessarily easy or even possible. Detection is the process of returning the equalized signal to discrete digital data to match the numbers originally coded and written to the channel. Figure 5.1 shows the processes of storage on a data channel from input to output data.

This section will start with a look at the channel and its capacity, go through channel compensation (linear equalization) and move onto detection, building up to the variable speed Bayesian detector.

5.1 Channel response

The 'channel' is the terminology used for what happens between a signal being recorded and the signal being recovered. If $w(t)$ is a recorded signal at time t and $r(t)$ is the replay signal from $w(t)$ then

$$r(t) = \mathcal{H}\left(w(t)\right) \tag{5.1}$$

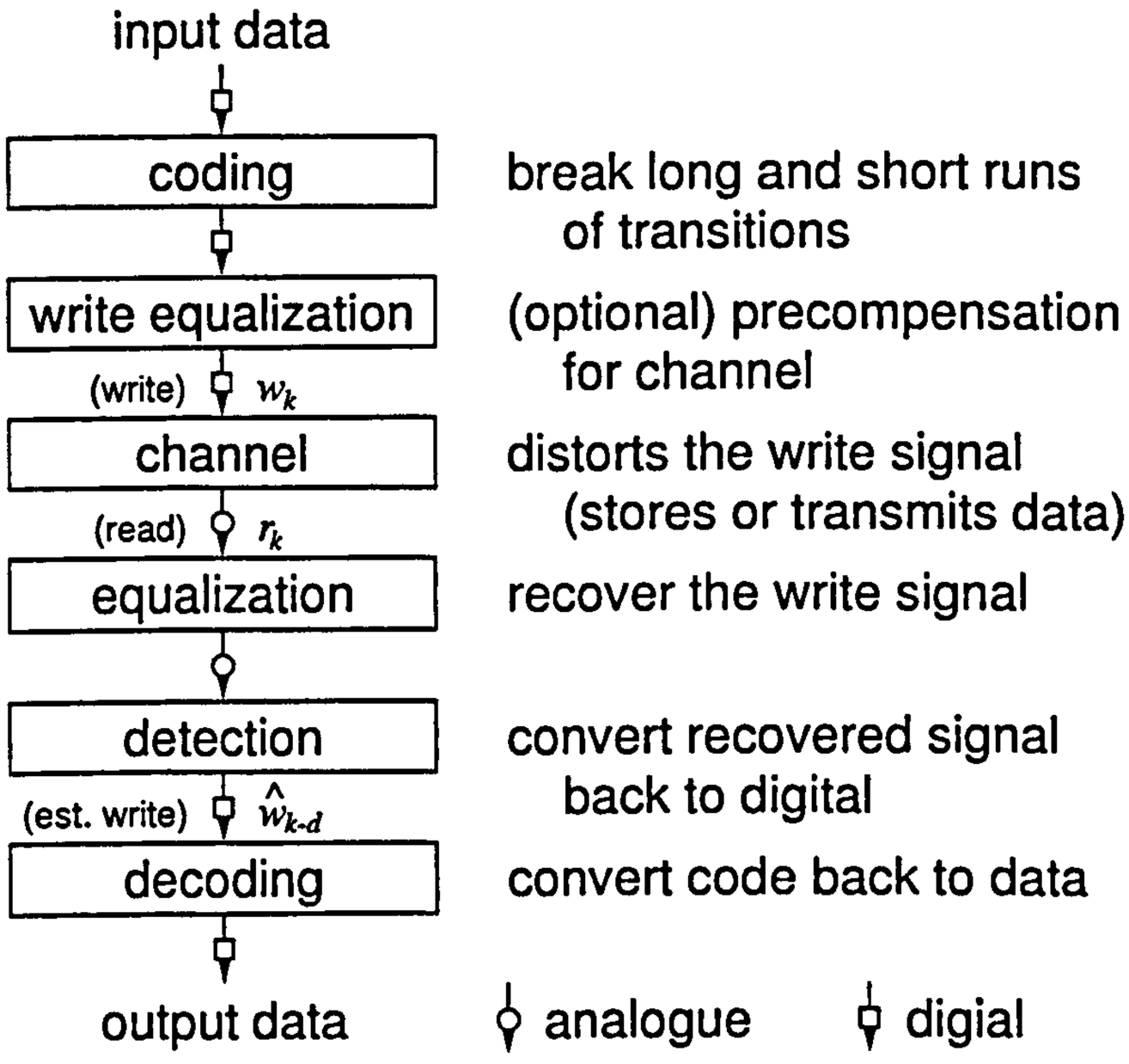


Figure 5.1: Process of storage in a data channel.

where $\mathcal{H}(\cdot)$ represents the channel operator. The channel can take on any form, but it is often approximated in fixed speed channels as a finite impulse response (FIR) or transversal filter which usually includes contributions from the write amplifier, write head, write loss, medium, read loss, read head, read amplifier and any post processing. (It does not have any noise contribution at this stage.)

FIR: finite impulse response

5.1.1 The Shannon capacity

Shannon [87] showed that a linear channel's information capacity, C in bits/s, can be derived from the channel frequency response.

$$C = B_w \log_2 (1 + \text{SNR}) \tag{5.2}$$

where B_w is the bandwidth defined by -3 dB markers and SNR is the signal to noise ratio.

By splitting the frequency spectrum into sub-bands, it is straightforward to show that equation (5.2) can be expanded into

$$C = \int \log_2 (1 + \mathrm{SNR}(f))\ df \tag{5.3}$$

to give the absolute maximum channel capacity as an integral over frequency, f.

Although the magnetic recording channel is not linear, the replay side is. (The loss factors are due to separation, the head gap, electronic band limits, and so on.) However it is not valid to record white noise onto the medium, but a white binary signal does have very similar properties to classic white noise. A white binary signal is the convolution of a top-hat function with a train of impulses of unit size with random sign. The power spectrum of this signal is the product of a wide sinc function and a white background. The simple magnetic recording model suggests that the medium will record only the sign of the magnetic field: since the write magnetic field is always in saturation the medium will also become a saturated white binary signal. However, the frequency characteristics of the recorded signal may be different due to the grain size, but it remains white, with a different top-hat width characteristic. There are complications due to finite field rise and fall times from the head and electronics, but this effect is minimal (and is ignored) at the densities encountered in this study. So, at least in theory, the non-linear write channel can be treated as linear when used with a white binary driver, and its frequency response (given a perfect reader) is the product of a sinc-like function (from the granularity of the medium) and the recorded white signal. For modem-like recording on an analogue (biased linear) magnetic channel, the amplitude modulation index must also be factored into the calculation [51]: this makes the Shannon limit strictly an upper bound of capacity. It must also be recognized that in the upper limit of grain size (where the write-side loss is greater than the read-side loss) the recovered pattern is now totally dependent on the grain layout and positions: the signal's entropy reflects the grain entropy. Fortunately, the thickness of the medium and width of the head with the magnetic credit card ensures that

read losses dominate the write losses. The grain particle size only produces the write-side noise signal, while read side noise is generated by the head and electronics.

Thus the maximum bit storage capacity of, say, a credit card can be inferred from the channel frequency* response and the channel noise frequency response assuming that the (bit-density) responses are independent of the card velocity. (This is mostly true, but there are band limitations in the read amplifiers and a fixed sampling rate that do make a small contribution to the response.)

5.1.2 Frequency, impulse and autocorrelation responses

A variety of methods are available to measure the SNR frequency response of a channel, as it is extensively used in all facets of communications engineering. Some straightforward methods are presented below.

Direct frequency measurement

A signal of known frequency is recorded onto the medium. The medium is then read back and the amplitude of the replay signal is noted.

There are two major drawbacks to this method. The first is time: only one frequency can be recorded at once, so to get a complete spectrum, many separate readings have to be taken. The second is due to non-linear effects: a sine wave recorded onto, say, magnetic tape will be recorded as a square wave which contains frequency components that were not in the original signal. It follows that peak amplitude readings may be misleading. At the high end of the frequency range, where the signal will be sine-like, the technique is more accurate.

This method is still useful for measuring the response of linear systems, notably biased tapes, where standard pre-recorded test signals can be used to give results that are easy and quick to interpret (especially if a set of frequency sweeps are recorded).

---

*The use of the words *frequency* and *spectrum* may be misleading because they are in fact *bit-density* and *bit-density spectrum*. In a constant velocity system the terms are equivalent and the former expressions will be used throughout the rest of this chapter.

Spectrum of an isolated impulse response

The Fourier transform of a Dirac delta function (an impulse) is a flat spectrum. Recording a set of impulses onto a medium results in a set of distorted impulses (or dipulses—the differential of an impulse) read back because it is not possible to physically record impulses. However a square wave (the integral of an alternating delta function train) can be recorded, and impulses (or transition responses) in the form of alternating approximate Lorentzian* curves (see section 5.1.3) are read back. The frequency transform of one of these (integrated) impulses gives the frequency response of the channel.

When read back, the recorded square wave may not have perfect positive and negative symmetry, although this is not a major setback. However, the square wave also sets the read head and circuitry into its full range of amplification. There are finite rise times and slew rates for inductive devices and amplifiers, and a full scale change will take longer than smaller perturbations; giving an apparently reduced bandwidth. In addition, the DC portion of the spectrum is emphasized.

The individual transition responses from, say, a recorded square wave can give a useful map of frequency response as a function of time (the spectrogram of the channel).

Spectrum of a white binary signal

The Fourier transform of white binary noise approximates, over time, to a flat frequency spectrum (with a wide sinc envelope). Injecting noise into the channel and measuring the spectrum of the output noise gives a direct reading of the channel response.

On a magnetic channel the non-linearities of the medium can lead to peculiar distortion effects in the spectrum. It is also worth noting that the spectrum

---

*The Lorentzian curve is the theoretical time response to a step function when read back through a head with an ideal though finite gap. The Lorentzian read-back signal, r_L, follows a gently curved shape: $r_L(t) \propto \frac{\pm a^2}{a^2+t^2}$. The parameter a represents half the PW_{50} of the curve: the pulse width at 50% amplitude.

derived is a time average over the noise period.

This method is most often used on radio channels and passive filter circuits that are almost completely linear.

Autocorrelation of a white binary signal

In a very similar method to the previous one, a known random (but not necessarily white) binary signal is written. When it is read back, the correlation of that signal with the signal recorded gives the channel's impulse response.

In order to avoid non-linear artifacts [76, 91] showing in the impulse response, Golay complementary sequences can be recorded on the channel [11]. These are constrained-length data streams that have perfect cross-correlation properties.

There is a major problem with this method in that it requires exact temporal synchronization between the written and retrieved signal. In this application it is not feasible to do this to the tolerance required. A simpler method is to record an unknown white binary signal (white in the sense that its autocorrelation is impulse-like: $\langle w(\tau)w(\tau+t)\rangle_\tau = 0$ for $t \neq 0$). The autocorrelation of the read-back signal is then the autocorrelation of the impulse response.

w = *written signal*

Noise spectrum

The noise spectrum per se usually includes head, media, environmental and electronic noise. The write system can be made to contain a very small proportion of noise, so this is usually ignored. The simple method to measure the channel noise spectrum is just to analyze the spectrum of a blank (or thoroughly erased) channel. This is often good enough, although it does not account for some of the dynamic features of the noise. (For instance, when the medium is magnetically saturated the noise may be dominated by the particle coercivity variation, while at lower magnetization levels impurities may be dominant.)

5.1.3 Measured frequency responses

The following charts show the frequency (bit-density) response of a low coercivity (~300 Oe) credit card, measured on the equipment described in chapter 2 using some of the methods previously described. The card was read at 4.5 or 9.7 in/s in the mechanical reader mechanism using the audio compact cassette inductive read head.

Impulse spectrogram

A low frequency square wave was written to a card, and the frequency response of each impulse (technically a transition response—the response of the channel to a magnetic state transition—an example is shown in figure 5.2) was plotted to form the spectrogram in figure 5.3. Notice that the bandwidth of the card is not constant along its length. There is a distinct drop in bandwidth just over a third of the way along and an increase in bandwidth towards the end of the card. The dip is probably caused, in this case, by the head lifting from the card—either from head tilt or warped plastic or perhaps even from dynamic effects when the head hit the card edge as it went past. The high frequency, low amplitude noise is attributable to the 8-bit quantization of the signal, electronics and domain noise.

It should also be noted that the apparent bandwidth from this method is significantly lower than that measured by other methods; partially because of the considerable low frequency content but mostly because the current slew-rate of the head depresses the frequency response of high amplitude signals.

Average spectrum of random signals

A high frequency random binary signal was written to the card. (Random in this case meaning that either forward or reverse current was forced through the write head, the choice being determined by a pseudo-random sequence.) The write clock frequency was chosen to be far higher than the anticipated response of the channel, so that the band-limiting properties of the channel disguise the

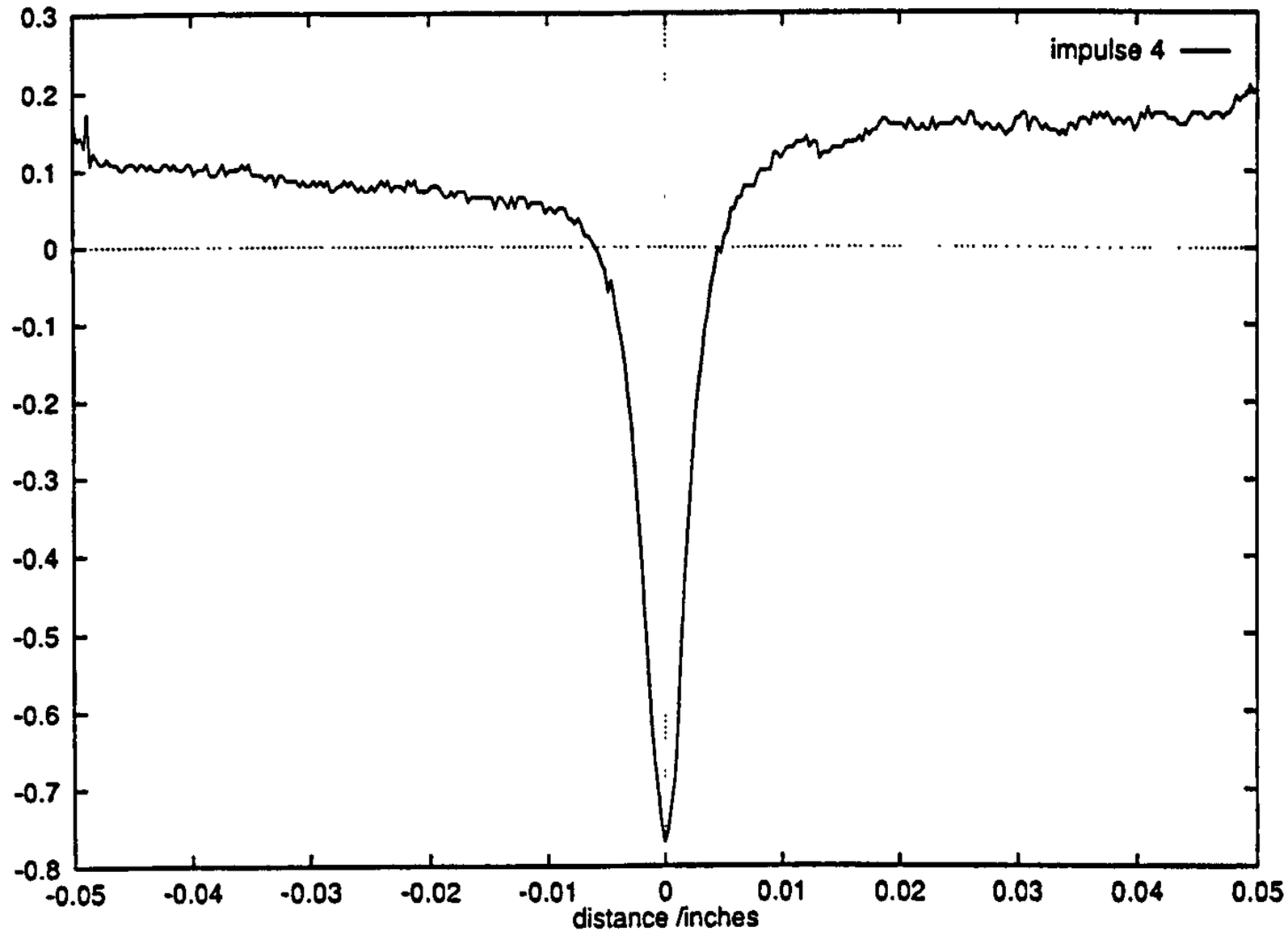


Figure 5.2: Isolated transition response (half-dipulse) measured from a 300 Oe (lo-co) credit card (inductive head).

fact that the written noise is also bandlimited. A small sample of retrieved signal is shown in figure 5.4.

The spectra of four retrieved random signals averaged along the whole card length are shown in figure 5.6. The chart shows that there is no particular difference between using a random write signal with a clock at 25 kHz and 50 kHz (the card speed was about 4.5 in/s), as would be expected. There was a marginal advantage in using pulsed writing at 0.25 duty cycle (the technique is illustrated in figure 5.5, pulsing gives a better temporal coherence to the critical magnetization field, H_c). The difference was not great however, and there is little that can be done on the write-side of the credit card to improve matters on the read-side.

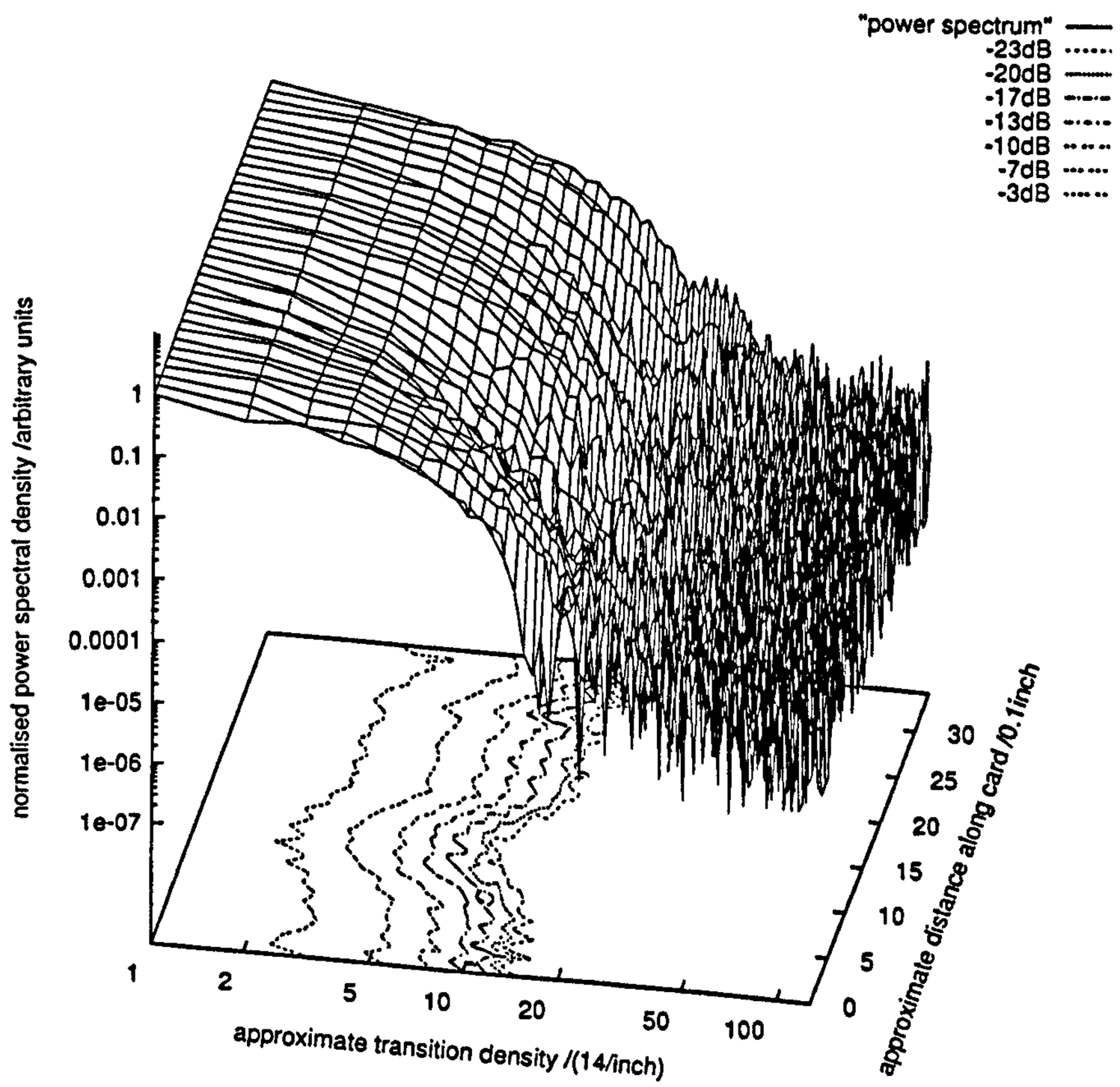


Figure 5.3: Transition density spectrum along a card measured from isolated impulses (inductive head).

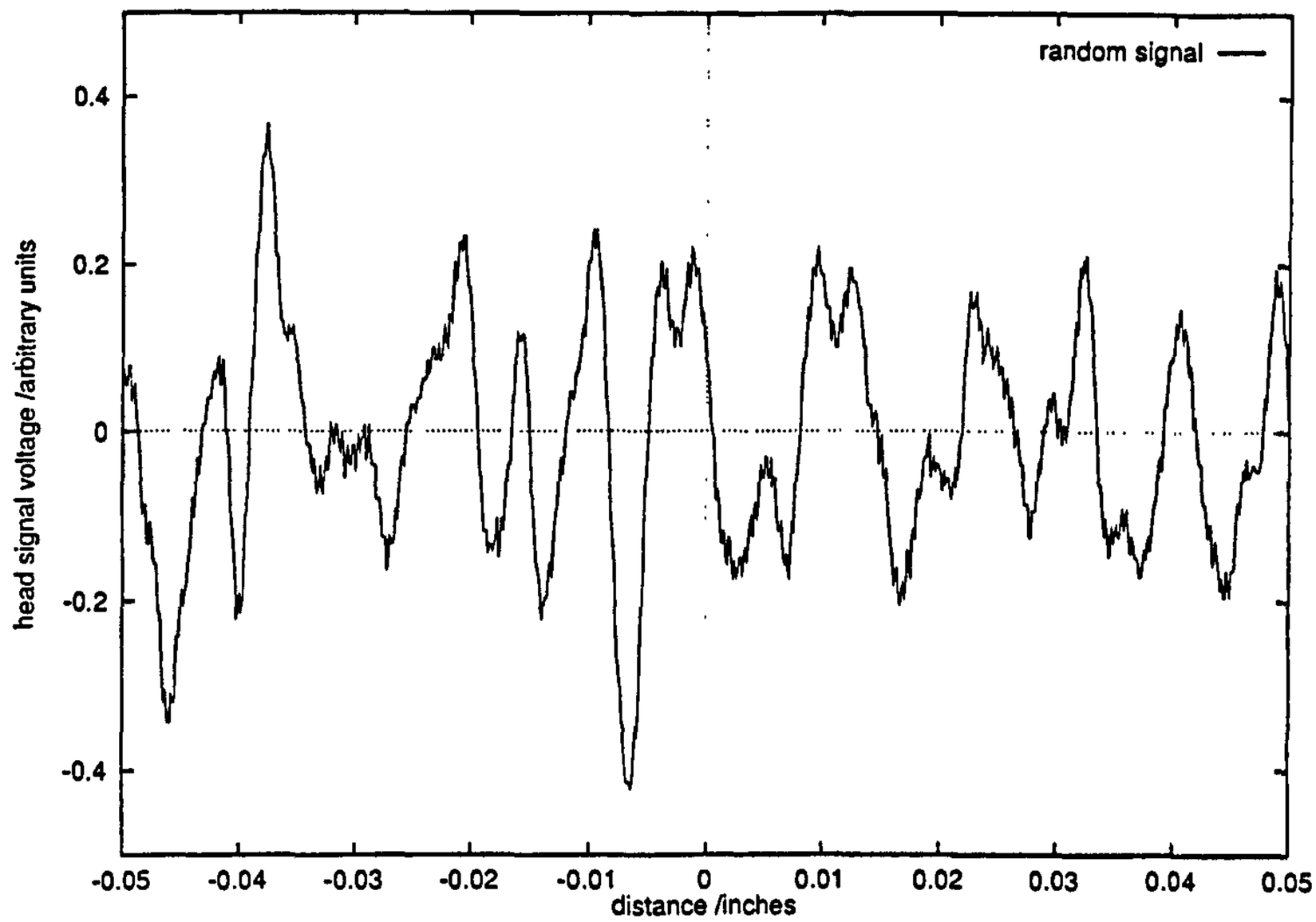


Figure 5.4: Read-back signal from random write signal (inductive head).

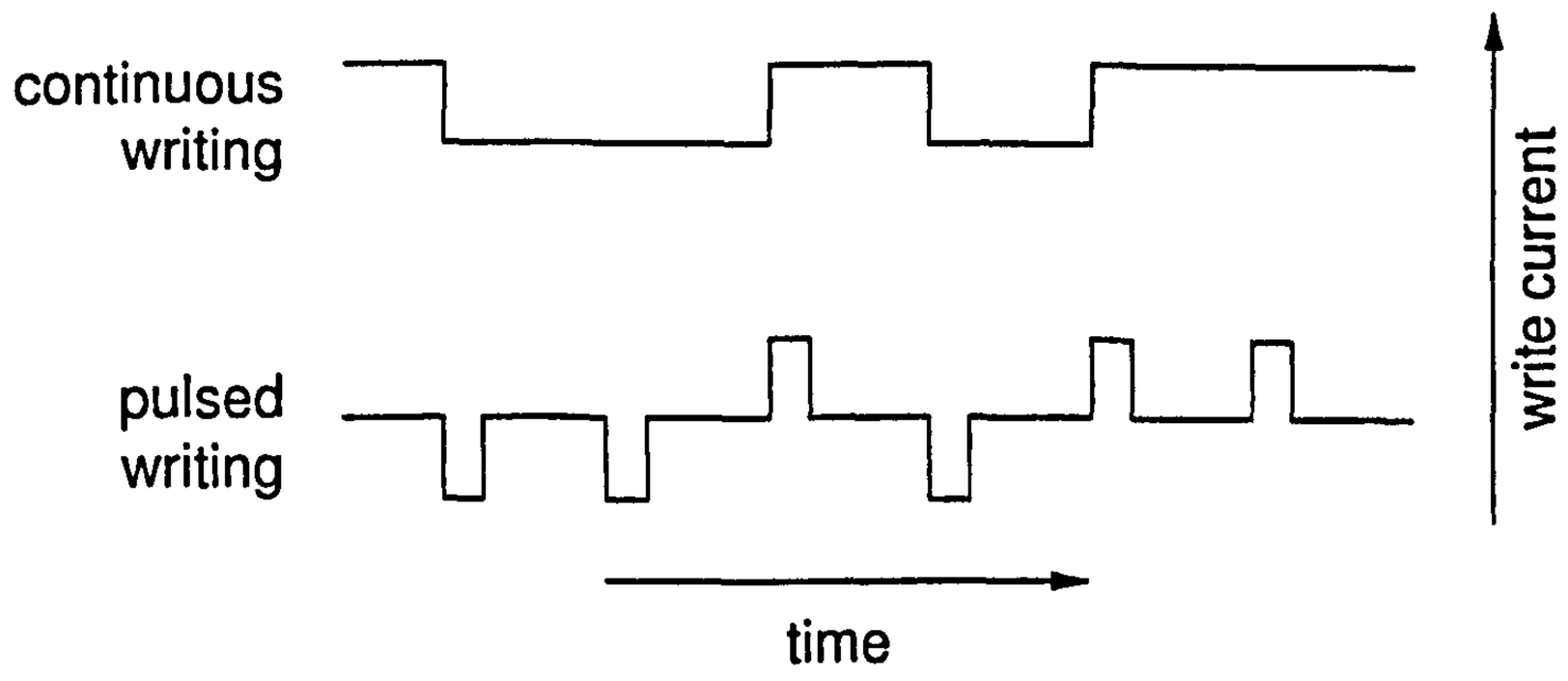


Figure 5.5: Difference between pulsed writing (0.25 duty cycle) and continuous writing.

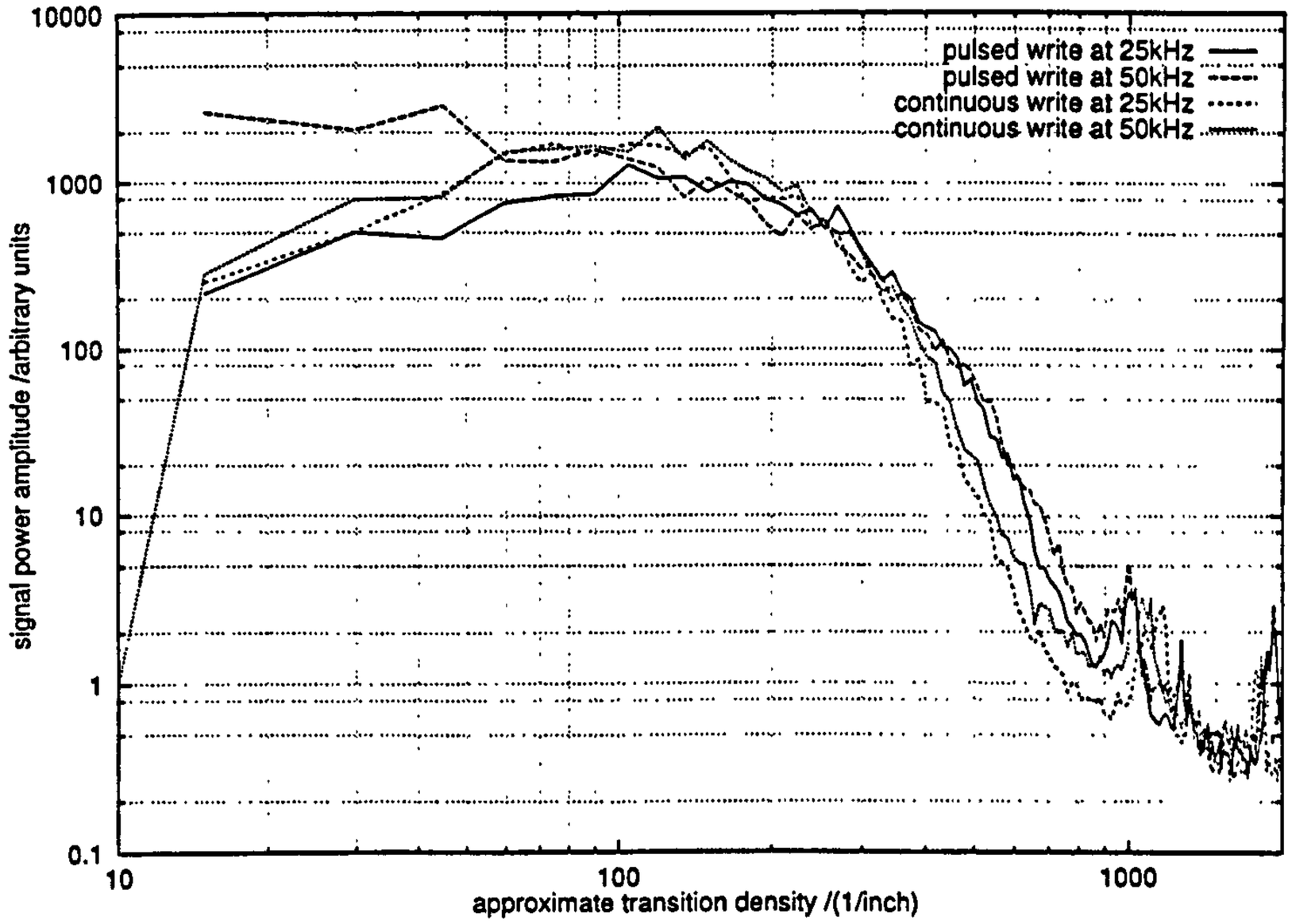


Figure 5.6: Readback spectra from random write signal (inductive head).

Autocorrelation of random signal

The autocorrelation of a retrieved random binary signal was calculated and the non-zero portion is shown in figure 5.7. (Compare this with the signal traces in figure 5.24 to inspect the user densities encountered.) This impulse autocorrelation compares very favourably with the expected shape of the dipulse autocorrelation for an inductive read head. The transition response (figure 5.2) should, theoretically, take the shape of a Lorentzian [9, 63], i.e.,

$$r_L(t) \propto \frac{\pm a^2}{a^2 + x^2} \tag{5.4}$$

where x is distance and a^2 determines the width (PW_{50}) of the pulse. The impulse response is the differential of the transition response. The autocorrelation of the derivative of the Lorentzian is shown in figure 5.8: this shape is also similar to the second derivative of the Lorentzian, but the first differential is appropriate for this system.

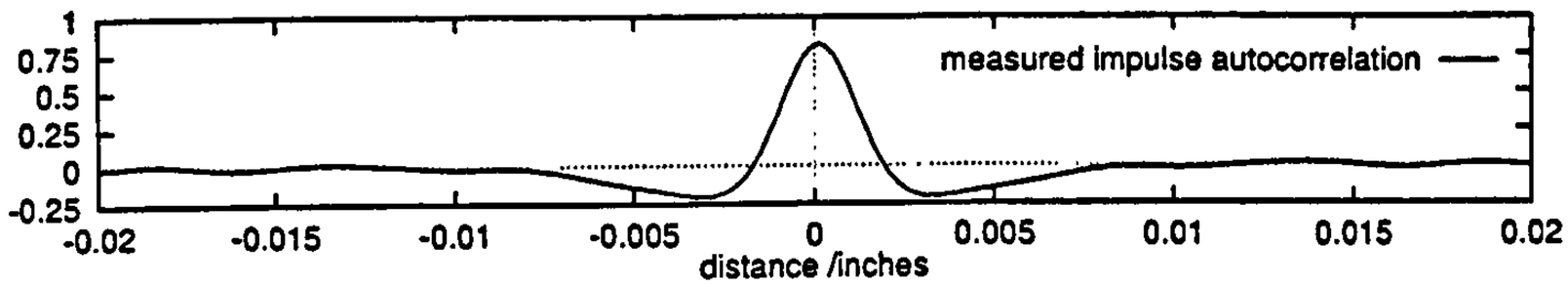


Figure 5.7: Measured autocorrelation of impulse response.

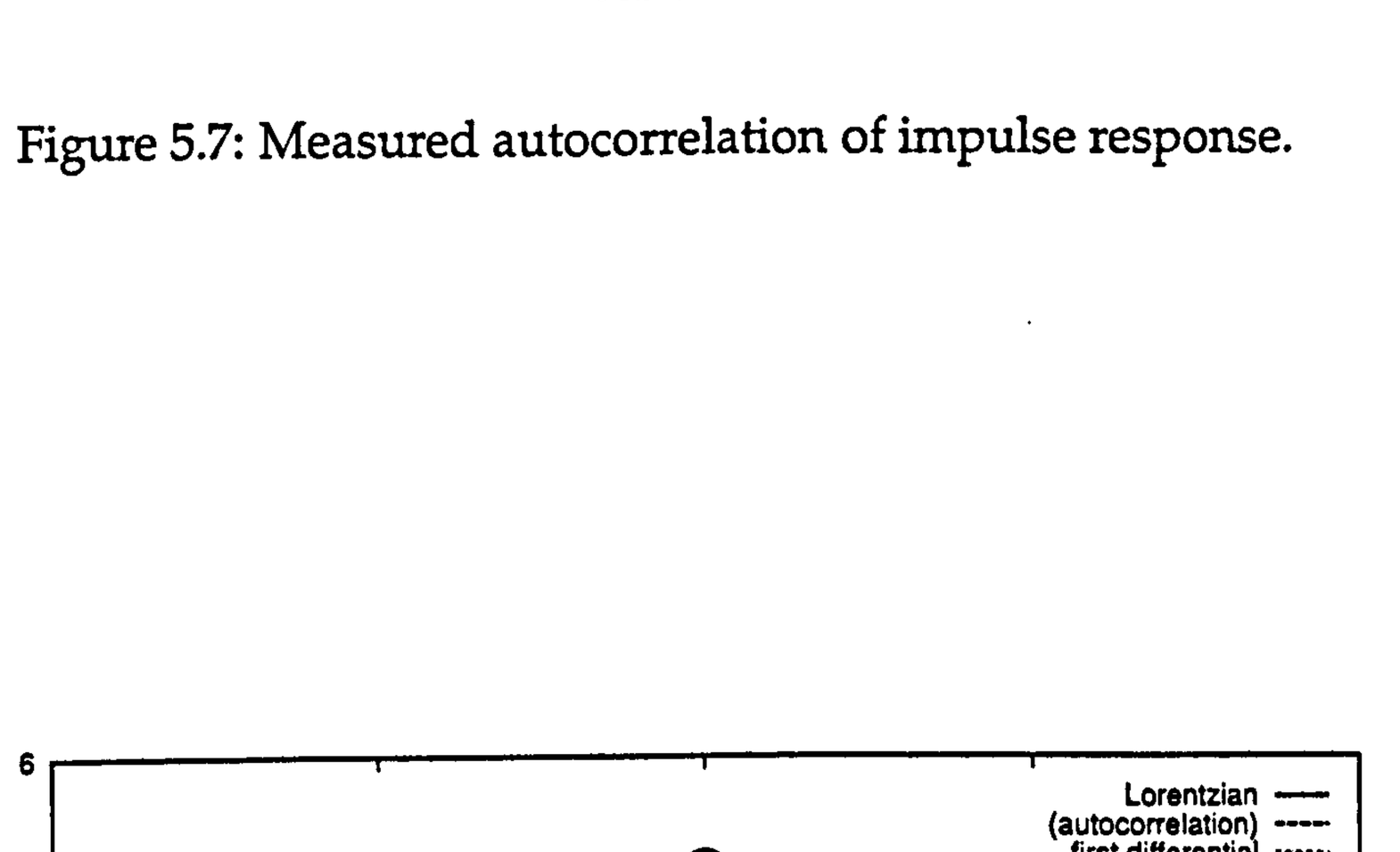


Figure 5.8: Lorentzian function and its derivatives with autocorrelation.

5.1.4 What is the theoretical storage capacity of a credit card?

Figure 5.3 shows a fairly clear noise floor with some systematic electronic noise. It is known that wider read heads have better SNR characteristics than narrower heads (because of the constructive averaging effect of the wider head over the track). However, it is expected that the system should have a noise floor lower than −60 dB for SNR from the head in a high density particle tape system, −40 dB of quantization noise and probably around the same for electronic noise. The observed noise floor is approximately −30 dB in the regions of interest, and the discrepancy is most likely caused by a combination of particle sizes and non-magnetic noise. It would be over-optimistic to think that real-life (retail point of sale) noise figures would be any better.

SNR: signal to noise ratio

Taking a constant noise floor of −30 dB (a value of 2.0 in figure 5.6) and using this in equation (5.3) gives an upper limit raw bit capacity of around 4500 bits per inch (bpi) or just under 2 kBytes per track. If the tracks were perfectly aligned with the read head at the optimum track pitch (24 min—the same as the head width) then over 20 kBytes could be stored in every card. No (constant speed) measurements were made with the MR (DCC) head, but disregarding track pitch practicalities (section 2.4.1) the theoretical storage capacity is in the hundreds of kilobytes.

bpi: bits per inch

min: milli-inch

MR: magneto-resistor

DCC: Digital Compact Cassette

Of course, it is not currently possible to achieve this theoretical density in practice. An immediate factor of (less than) two comes from the velocity problem—there are two degrees of freedom at every point: one of signal value and one of velocity, meaning some density must be sacrificed to allow clocking information. The highest useful recording density on the card with an inductive head has been found to be about 1500 bpi (figure 5.35); a factor of three short.

5.2 Channel compensation

If the channel function is represented by $\mathcal{H}(\cdot)$ as in equation (5.1) then the read signal, $r(t)$, when operated on by $\mathcal{H}^{-1}(\cdot)$ will give back the original coded signal.

Two problems present themselves. Firstly, $\mathcal{H}^{-1}(r(t))$ is not (necessarily) unique or determinable. Second, given a write signal, $w(t)$, a better channel model is

$$r(t) = \mathcal{H}(w(t)) + n(t) \tag{5.5}$$

where $n(t)$ represents noise which is random and non-determinable. Now the expression $\mathcal{H}^{-1}(r(t))$ becomes merely an approximation to $w(t)$, and may in fact be dominated by the noise contribution to the signal.

$\mathcal{H}^{-1}(\cdot)$ is still a reasonable first choice for equalization, but the noise in the estimate of the original signal,

$\mathcal{H}$ = *channel operator*
n = *noise signal*
r = *read signal*
w = *written signal*
$\hat{w}$ = *write estimate*
e = *error*
t = *time*
f = *frequency*

$$\hat{w}(t) = \mathcal{H}^{-1}(r(t)), \tag{5.6}$$

means there will be a (perhaps considerable) error, e, between the deconvolved signal and the originally written signal.

$$e(t) = \hat{w}(t) - w(t) = \mathcal{H}^{-1}(\mathcal{H}(w(t)) + n(t)) - w(t) \tag{5.7}$$

SNR: signal to noise ratio

It is now clear why the Shannon capacity, equation (5.2), is reliant on the SNR to give the maximum decodable information: the channel signaling rate is unlimited, but when $e(t)$ causes decoding errors due to noise $\mathcal{H}^{-1}(\cdots + n(t))$ in equation (5.7) the usable data rate must be decreased. (If $n(t) = 0$ then $e(t)$ would be zero also, with this (possibly unrealizable) $\mathcal{H}^{-1}$ function.) Indeed, the nature of the problem enforces that the faster the signaling rate, the more sensitive $\mathcal{H}^{-1}$ is to noise.

5.2.1 Equalization

Consider the channel model shown in figure 5.9. If $\mathcal{H}(\cdot)$ is known (or approximately known) to be a convolution (which can be converted to a frequency limiting filter), then the inverse filter (which can also be written as a convolution) will perform equalization. Figure 5.10 shows a simulated example of the

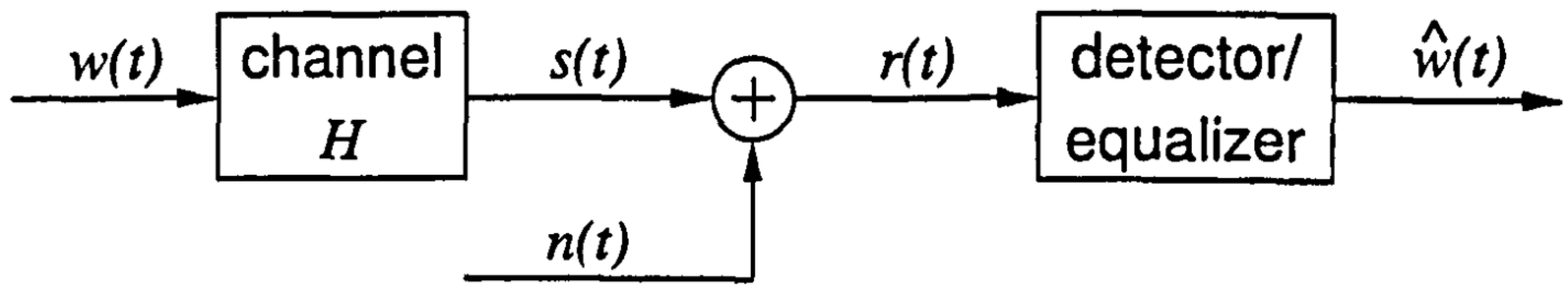


Figure 5.9: Channel model and terms for equalization.

inverse filter algorithm described below.

First a useful (theoretical) signal, the noise free read-back signal, $s(t) = \mathcal{H}(w(t))$ shall be introduced. Now using the notation $x(t) \Longleftrightarrow X(f)$ to mean *time domain* $\Longleftrightarrow$ *frequency domain* and approximating $\mathcal{H}(w(t))$ to a convolution, $h(t) * w(t)$, equation (5.5) can be written

s = noise free r

H = channel transfer function

$$r(t) = h(t) * w(t) + n(t) \Longleftrightarrow R(f) = H(f)W(f) + N(f) = S(f) + N(f) \quad (5.8)$$

and hence derive in the case $N(f) = 0$

$$\hat{W}_n(f) = \frac{R(f)}{H(f)} = H^{-1}(f) \cdot R(f) \Longleftrightarrow \hat{w}_n(t) = h^{-1}(t) * r(t). \quad (5.9)$$

Figure 5.10 clearly shows that this naïve inverse filter is insufficient to reconstruct the original signal. It is (almost) entirely due to the noise in equation (5.5) being amplified out of all recognition. In the simulated data, even just the quantization and sampling noise (less than 0.5% of the peak signal) is excessively disruptive.

A modified method, called optimal or Weiner filtering, gives a much better recovered signal if the noise can be estimated. Following the derivation in [80], *another* filter, $\phi(t) \Longleftrightarrow \Phi(f)$, is included during the inverse filter operation,

ϕ = Weiner filter

$$\hat{W}_W(f) = \frac{R(f)\Phi(f)}{H(f)}, \quad (5.10)$$

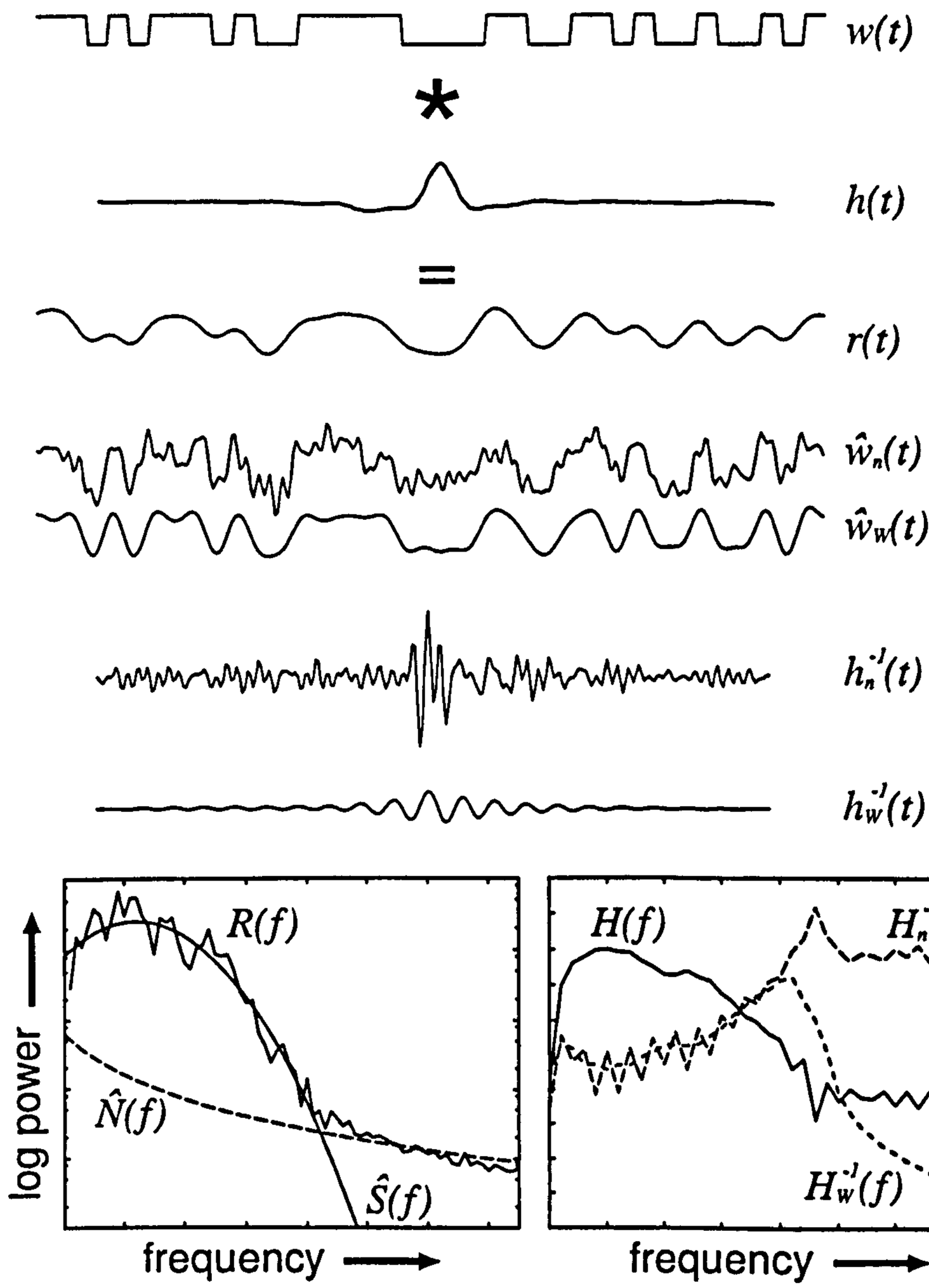


Figure 5.10: Example inverse filter system: $w(t)$ is the original digital signal. After passing through the channel $h(t)$ it is corrupted into $r(t)$. (The response $h(t)$ shown here was measured from a real credit card, $r(t)$ is a simulation: simply the convolution $w(t) * h(t)$.) A naïve inverse filter is shown as $h_n^{-1}(t)$, it is derived from the inverse of $h(t)$. Below, $h_W^{-1}(t)$ is the optimal filter derived from estimates of the signal ($\hat{S}(f)$) and noise ($\hat{N}(f)$) present in $R(f)$. Both inverse filters are similar at low frequencies, but at high frequencies, where noise is dominant, the optimal filter depresses the signal. The equalized results from the two inverse filters, $\hat{w}_n(t)$ and $\hat{w}_W(t)$, are shown with $r(t)$ for comparison. However, $w(t)$ can never be fully reconstructed with a simple inverse filter.

and is chosen to minimize the noise-induced mean squared error.

$$\left\langle |e(t)|^2 \right\rangle_t = \left\langle |\hat{w}_W(t) - w(t)|^2 \right\rangle_t = (\text{constant}) \left\langle \left| \hat{W}_W(f) - W(f) \right|^2 \right\rangle_f \tag{5.11}$$

The last equality is from Parseval's theorem. Using the definitions of $\hat{W}_W(f)$ and $W(f)$ above and the *definition* that the signal and noise are uncorrelated ($\langle |N(f)\cdot S(f)|^2 \rangle_f = \langle |N(f)|^2 + |S(f)|^2 \rangle_f$), the following relationship is evident

$$\begin{aligned} \left\langle \left| \hat{W}_W(f) - W(f) \right|^2 \right\rangle_f &= \left\langle \left| H^{-1}(f) \left\{ (S(f) + N(f))\, \Phi(f) - S(f) \right\} \right|^2 \right\rangle_f \\ &= \left\langle |H(f)|^{-2} \left\{ |S(f)|^2 \, |1 - \Phi(f)|^2 + |N(f)|^2 \, |\Phi(f)|^2 \right\} \right\rangle_f \end{aligned} \tag{5.12}$$

To find the minimum mean squared error (MMSE) with respect to Φ the stationary point(s) of this equation must be determined. The minimum solution for equation (5.13) occurs when the inner equation is minimum for every point on f, hence

$$\frac{d}{d\Phi(f)} |H(f)|^2 \left\{ |S(f)|^2 \, |1 - \Phi(f)|^2 + |N(f)|^2 \, |\Phi(f)|^2 \right\} = 0 \tag{5.13}$$

$$\Phi(f) = \frac{|S(f)|^2}{|S(f)|^2 + |N(f)|^2} \tag{5.14}$$

This real filter, $\Phi(f)$, requires that both the noise-free power spectrum, $|S(f)|^2$, and the noise spectrum, $|N(f)|^2$, are known. It is also gratifying that in the case of no noise, $\Phi(f) = 1$ and equation (5.10) simplifies to equation (5.9).

In the case of a time stationary channel, Weiner filtering is a reasonable proposition. In the variable channel case, it is not so useful, as $S(f)$ is neither constant nor easy to calculate or measure.

FIR: finite impulse response

Most common equalization methods are based on FIR schemes. There are many other systems from the image processing arena, often based on the greater redundancy of two dimensional signals. Examples include maximum entropy Bayesian image deconvolution [88] and wavelet spectrum completion [62]. In one dimensional systems their usefulness is diminished somewhat.

All these equalization methods rely on the channel being known and unchanging. This makes them unsuitable for credit card recording in their standard form.

5.2.2 Adaptive equalization

Recall equation (5.8), the expression giving the signal read back from a channel. Weiner filtering requires that the signal, $s(t) = h(t) * w(t)$, and the noise, $n(t)$, are known, or estimatable in the frequency domain. Suppose $h(t)$ is unknown but $s(t)$ and $w(t)$ are known: can $h(t)$ be derived? The answer is yes, given time, it can be approximated using an adaptive filter.

In fact, the adaptive filter method will also average out the noise given $w(t)$ and $r(t)$, i.e., the original signal written and the signal read back; although it requires that the noise is uncorrelated and independent of the signal. This is very often the case with linear communications channels (modem, radio), where such noise is termed independent identically distributed (iid), but in magnetic recording the assumption only holds approximately true. The noise amplitude is related to the magnetic signal amplitude: equation (5.8) is not precisely correct in a magnetic channel. The degree of divergence of a magnetic channel from an ideal channel is dependent to a large extent on the system: for a magnetic card it is sometimes severe.

Because feedback systems can become unstable when placed in unknown environments, nearly all adaptive filters are based on FIR (feedforward) filters rather than the more general class of infinite impulse response (feedback) filters. Some environments can tolerate these IIR recursive adaptive filters [21], but their use is specialized.

FIR: finite impulse response

IIR: infinite impulse response

The most common FIR adaptive filter scheme uses the least mean squares (LMS) algorithm to update its filter taps. The precise derivation of this method is lengthy (see the first half of [40]) but a less formal method also gives the same result, and is described below.

LMS: least mean squares

This derivation relies on the *observation* that if any tap in the inverse filter

is slightly off-optimal, the error magnitude will be increased; hence the taps of the inverse filter should be modified until the average squared error is a minimum—the least mean squares condition. For the working below, no noise will be added ($\boldsymbol{n} = 0$), time is discrete (and is denoted by a subscript, e.g., $w_i = w(iT)$ where T is the sampling period) and that the channel is being equalized with FIR filter h_j^{-1}. The error in this case is the difference between the reconstructed (inverse filtered) signal and the original signal that caused it.

$$\begin{aligned} e_i^2 &= \hat{w}_i - w_i = \left((\boldsymbol{h}^{-1} * \boldsymbol{r})_i - w_i\right)^2 \\ &= \left(\left(\sum_j h_j^{-1} \cdot r_{i-j}\right) - w_i\right)^2 \\ &= \left(\sum_j h_j^{-1} r_{i-j}\right)^2 - 2w_i \left(\sum_j h_j^{-1} r_{i-j}\right) + w_i^2 \end{aligned} \tag{5.15}$$

Now consider the construction of the graph of e^2 against h_j^{-1}. Figure 5.11 shows the form of this graph for e^2 against h_1^{-1} and h_2^{-1}. (The full figure would show a multi-dimensional concave hyper-surface.) The minimum error position will only be zero error if there is no noise in the system.

The problem is now reduced to a simple procedure. Given a set of values, h_j^{-1}, how can these be changed to reduce the value of e_i^2? On a surface where the error increases when any h_j^{-1} is off-optimal, the gradient of the squared error, ∇e_i^2, barring noise, points directly away from the optimal value of $\boldsymbol{h}^{-1}$. The LMS algorithm uses the scaled negative gradient of the e_i^2 surface to step down towards the minimum error position.

$$\nabla e_i^2 = \begin{pmatrix} \frac{\partial}{\partial h_1^{-1}} \\ \frac{\partial}{\partial h_2^{-1}} \\ \vdots \end{pmatrix} e_i^2 \tag{5.16}$$

$$\frac{\partial e_i^2}{\partial h_k^{-1}} = 2r_{i-k}\left(\sum_j h_j^{-1} r_{i-j} - w_i\right)$$

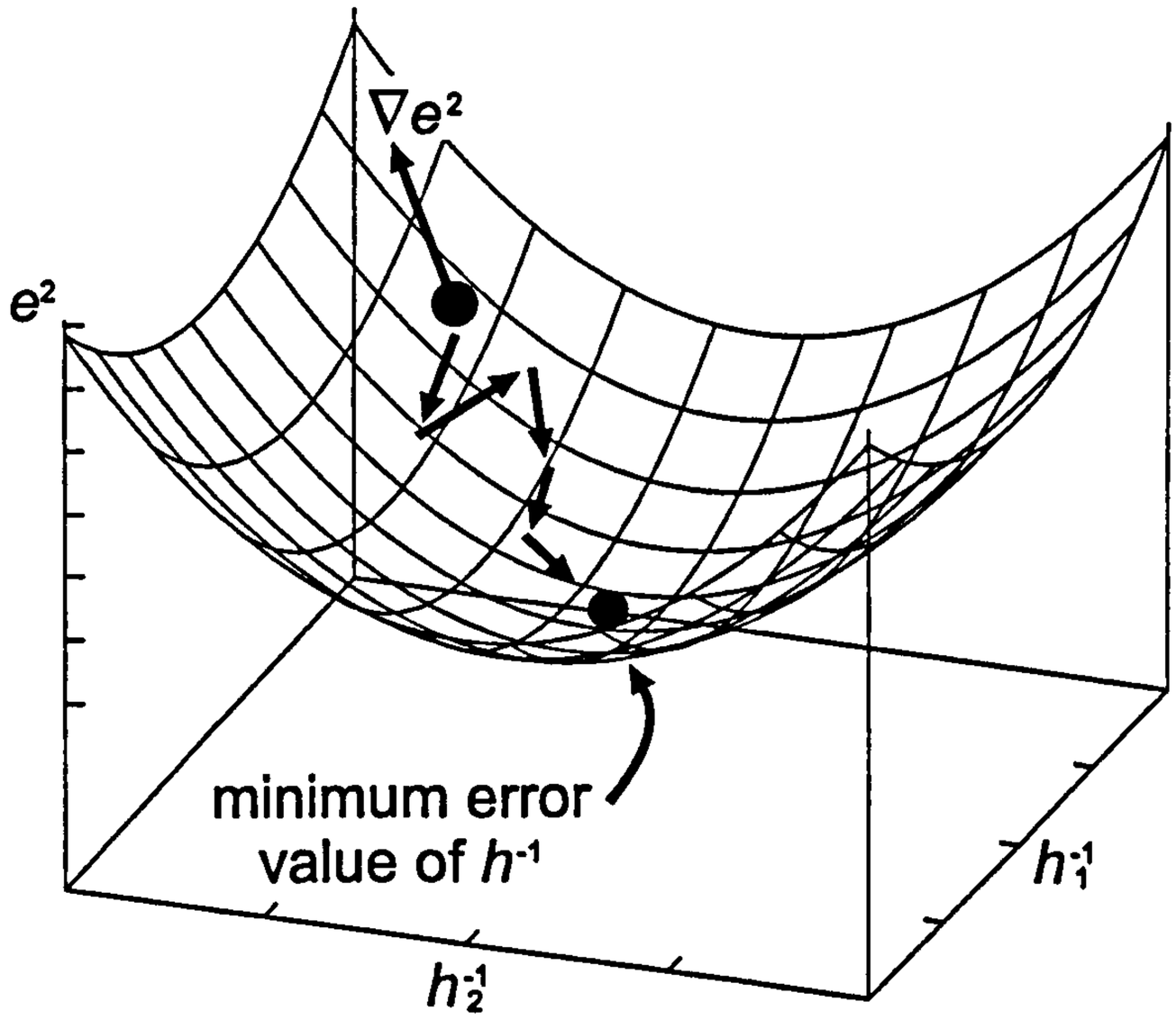


Figure 5.11: Cross-section of a multi-dimensional concave error surface with the path of a noisy $-\mu re$ marked, evolving towards the minimum mean squared error condition.

$$= 2r_{i-k}e_i \tag{5.17}$$

The gradient has physical units of e^2h so a scaling factor, conventionally $\frac{\mu}{2}$, must be used before subtracting (5.17) from h_j^{-1}. If this value is small enough, the vector $\boldsymbol{h}^{-1}$ will converge to the value giving minimum error without overshooting its target.

$$h_j^{-1} \longleftarrow h_j^{-1} - \mu r_{i-j}e_i \tag{5.18}$$

This update coerces the inverse filter to reveal more detail over time, and the progress of the filter taps is shown against the number of iterations in figure 5.12. In time, in a stationary channel, the LMS and optimal inverse filter taps will resemble each other, figure 5.13.

Equation (5.18) is the embodiment of the LMS updating algorithm. In practice, a channel must either be unknown and a known training sequence passed through it, so that the LMS filter taps can converge to the minimum error po-

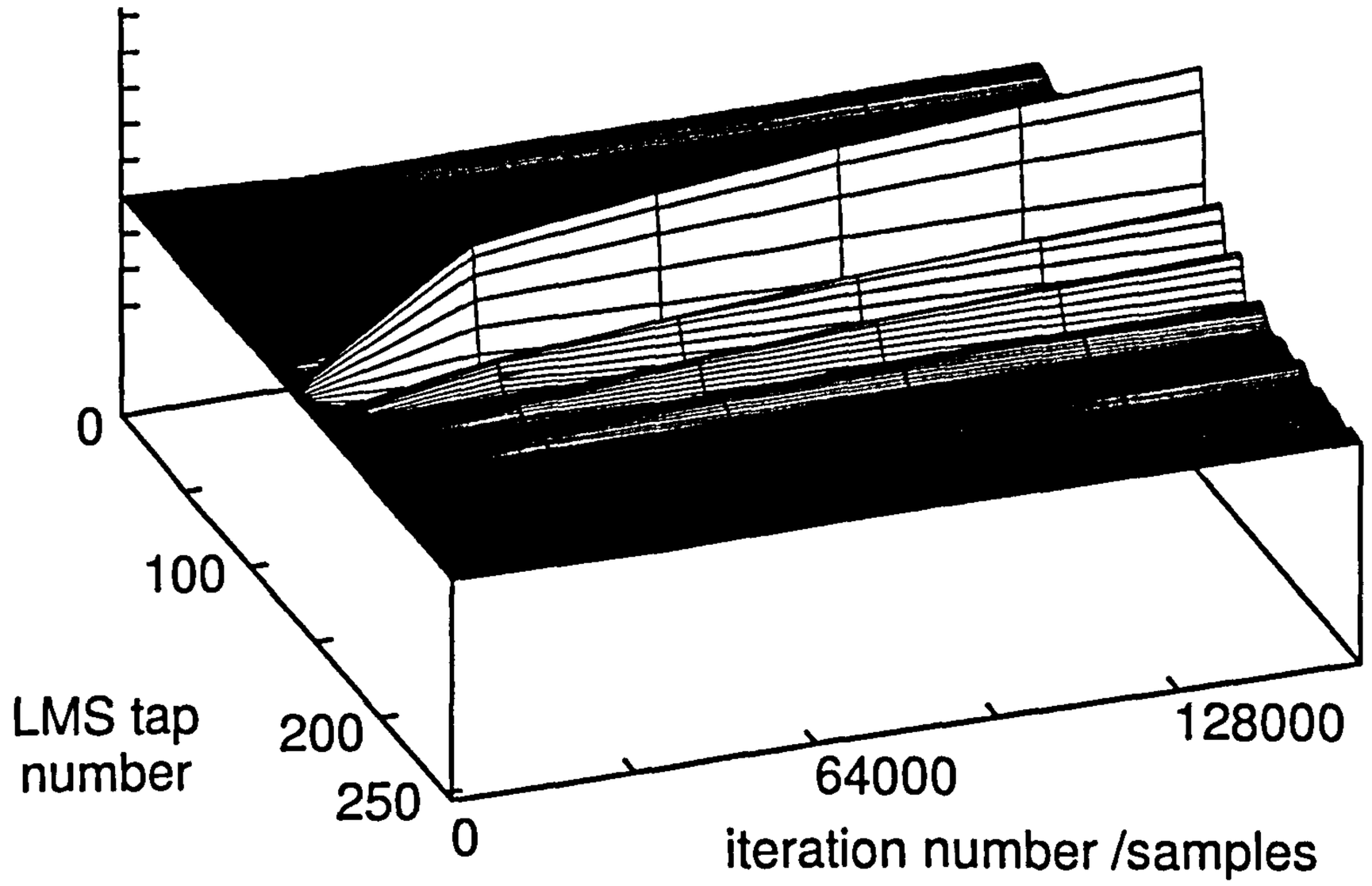


Figure 5.12: Evolution of inverse LMS filter taps.

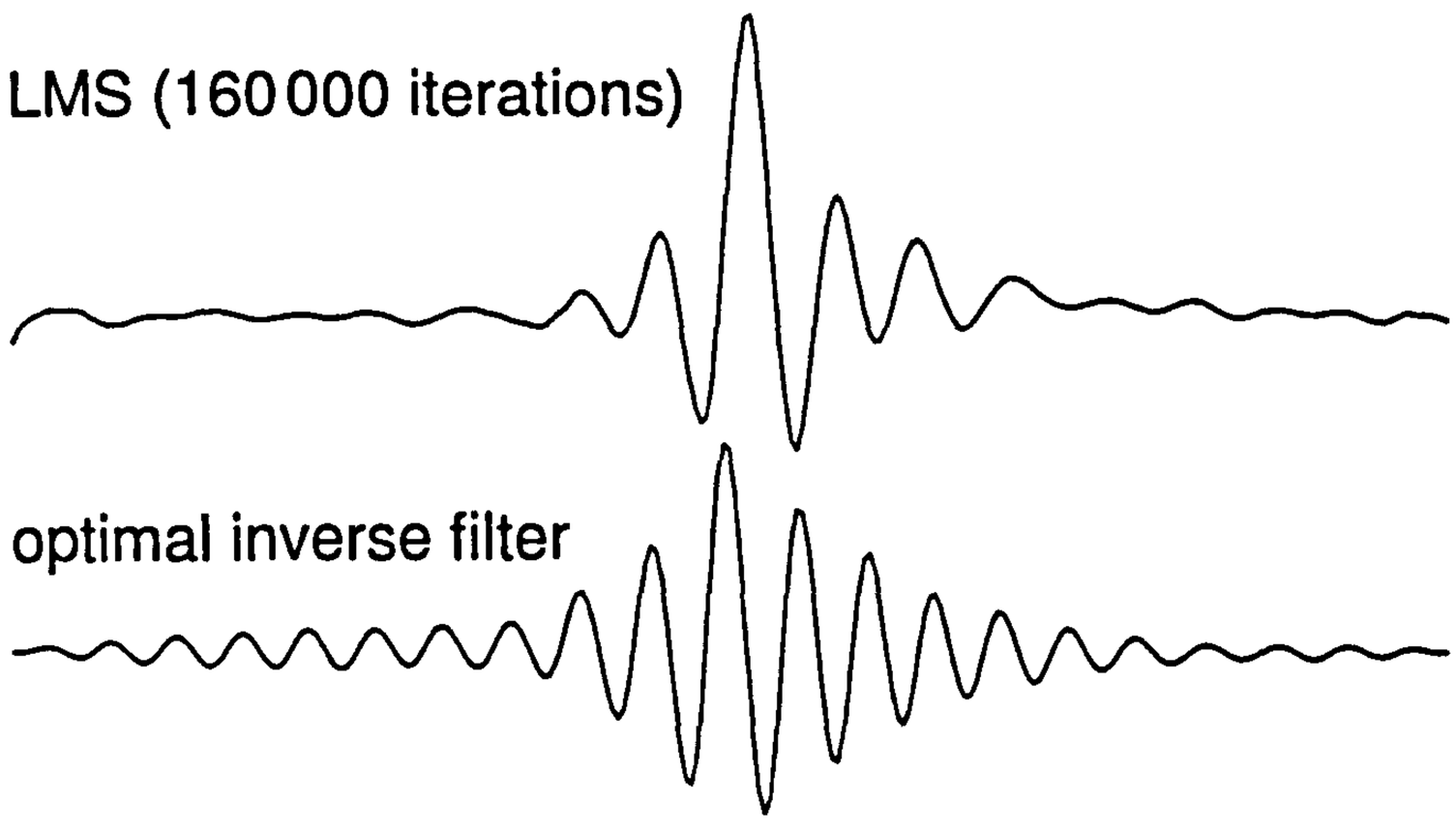


Figure 5.13: Comparison between the inverse filters for the channel shown in figure 5.10 and the LMS derived inverse filter.

sition, or the channel must be nearly known. In this latter case, it is common to employ decision directed equalization. This is the situation when, even with 'young' filter tap values, the detection device will produce mostly correct decisions. These values are fed back and used as the comparison signal (w_i in this notation). The iteration of taps is now statistically likely to converge to the LMS condition—although it is possible for h^{-1} to converge to $-h^{-1}$—and the time to convergence is data dependent. This structure is examined again in the form of the decision feedback equalizer (DFE) in section 5.4.4.

RLS: recursive least squares

A second common adaptive filter method uses a recursive least squared (RLS) error algorithm. The RLS filter evolves faster than LMS, basing its inverse filter estimate on all the data available so far. If the channel, based on the signal read back, is contradictory (for instance due to read errors) it can lead to instabilities in h^{-1}. If there are any errors in the training sequence this can be catastrophic. The situation is exacerbated further if the training sequence is derived from a decision feedback signal—there is an advantage in using adaptive filters that adapt slowly and do not quickly confuse themselves.

5.2.3 Non-linear adaptive equalization

The LMS adaptive filter algorithm is intimately related to FIR filters. The FIR structure is inherently linear (e.g., double the input signal's amplitude and the output signal's amplitude doubles). This means that the LMS algorithm, with its first-order update method and FIR structure, is also linear. Magnetic recording, on the other hand, is not linear (e.g., double the input amplitude and the output amplitude will probably *not* double). The usual solution to this is to modify the channel so that it becomes more linear. There is currently no general method available to adapt to non-linear channels, although there are some intermediate correction models.

The magnetic channel acts as a highly non-linear saturation medium. The write signal, $w(t)$, is usually digital, driving the magnetic medium into complete saturation. So, at least at fairly low densities, the channel magnetization

is a reasonable representation of $w(t)$ because it is always saturated (and hence also effectively digital). At very high density recording a phenomenon called non-linear transition shift occurs, whereby the magnetic fields from neighbouring bit cells interfere with that of the bit cell being written, so the magnetization pattern does not match $w(t)$. At low densities and low amplitudes, with the major signal degradation being from the perfectly linear finite-gap read head or head-to-medium separation, the magnetic channel can be considered a linear superposition of step functions of alternating polarity. So perhaps surprisingly, for digital magnetic recording, the non-linear medium appears fairly linear to the equalizer.

However when magnetic field strengths are high, the linearity of the magnetic transducer must also be examined.

Inductive heads can be designed so that their saturation magnetization is much higher than the field they will experience. The response of the head will be a symmetric, slightly saturating transfer function. Most inductive read head systems are equalized with linear filters.

MR: magneto-resistance
DCC: Digital Compact Cassette

Magneto-resistive (MR) heads are different. Take for example the Philips DCC read head. An MR head gives a saturating second-order resistance output corresponding to the magnetic field: that is the intensity, but not the polarity, of the field can be measured. Hence in the DCC head an external magnet (among other features [78]) is used to bias the head, so that the medium field added to the bias field yields the direction and strength of the medium field. However, the response is now not symmetrical (look back to figure 1.8, page 16). A large medium field in one direction, added to the bias field will saturate the resistance, but a reverse medium field will more than completely cancel the bias field and the resistance will then appear to 'flip-back' on itself. There is a hardware solution to the MR saturation and inversion problem, and this was discussed in section 1.1.8, page 14, although it cannot be applied with the DCC's narrow MR heads (those 9 heads share a common controllable electromagnetic bias current). A modeled distorted signal waveform is shown in figure 5.14 to illustrate the manifestation of the effect.

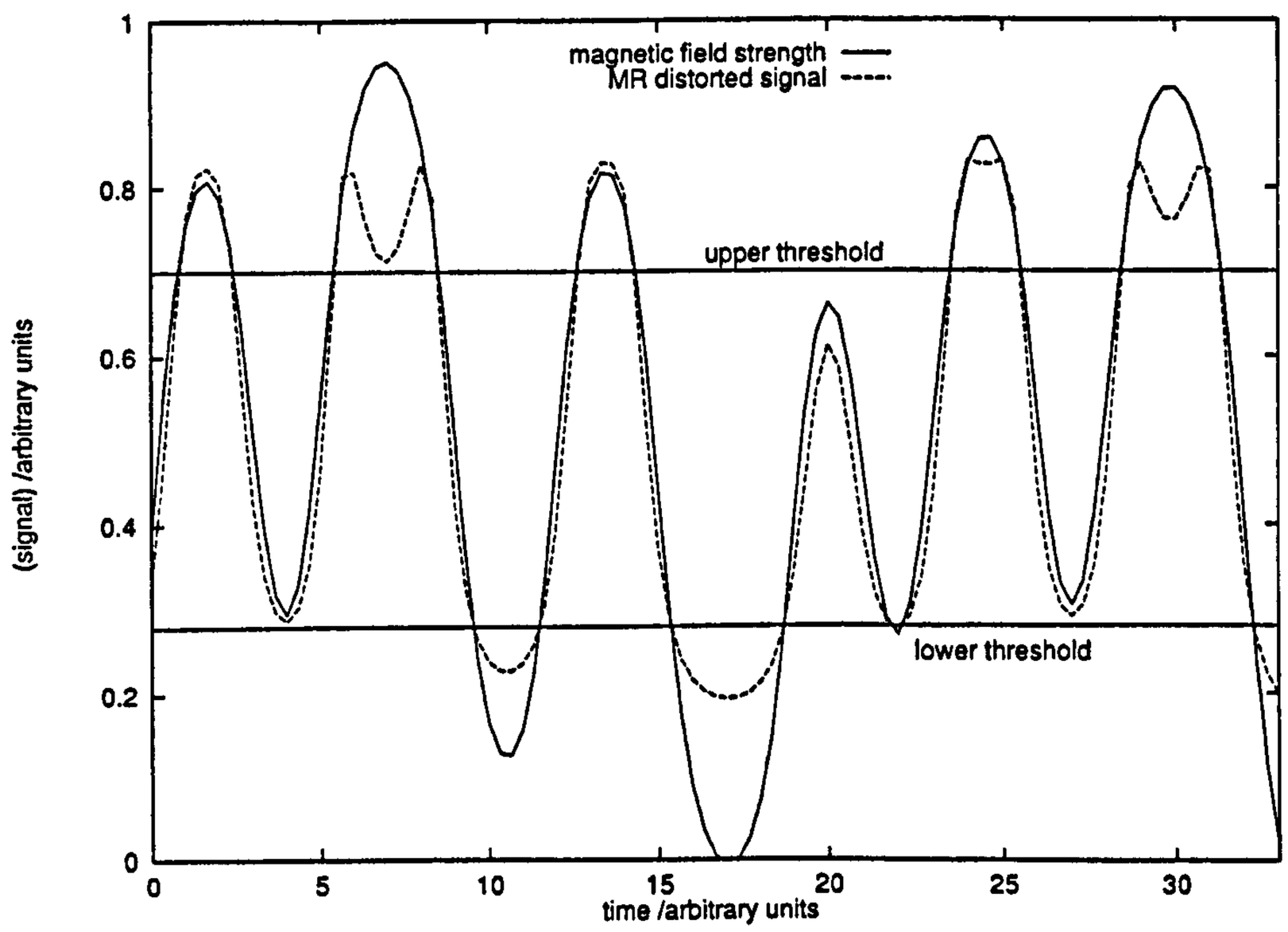


Figure 5.14: Modeled distorted magnetic read-back signal with non-linear approximate thresholds marked. The distorting function is $y \longleftarrow (\tan^{-1}(y))^2$, which resembles the MR distorting function.

Moderated error signal

If the write signal, w_i, was large, then the read-back signal, r_i, will be large and hence distorted: thus the squared error between the written and read signals, e^2, will be worse (larger) than it would be in the perfectly linear situation. In this case, there is merit in not updating h^{-1}. If it is known that the error signal, e_i, is due almost entirely to non-linear distortion then the error feedback can be softened. A suitable algorithm is simply not to update the filter when the read-back signal is larger than a certain threshold (an appropriate threshold is marked in figure 5.14). This is done in some radio channels when a noise burst is present [92]. A more specialized and sophisticated algorithm may note the onset of flip-back and try to correct the signal in the appropriate places, although this has not been attempted.

Direct (forced) linearization

b = *magnetic field*

β = *distortion function*

If the read-back magnetization level, $b(t)$, is transferred and distorted by a function, $\beta(\cdot)$, to the electrical read-back signal,

$$r(t) = \beta(b(t)), \tag{5.19}$$

then the linearized signal is simply the electrical signal after being operated on by the inverse transfer function, $\beta^{-1}(\cdot)$.

$$\hat{b}(t) = \beta^{-1}(r(t)). \tag{5.20}$$

There are three problems. The first is that $\beta(\cdot)$ must be measured: this may not be feasible if, for instance, if is expected that $\beta(\cdot)$ will change often although in situ methods do exist [101]. Another problem is that the noise is also modified by $\beta^{-1}(\cdot)$ which may cause problems for the equalizer later on. The final, and most fundamental, problem is that $\beta^{-1}(\cdot)$ may not be single valued, as is the case with MR flip-back.

Direct linearization is regularly performed in analogue electronics where,

say, a logarithmic diode transfer function can be linearized by an exponential transistor transfer function. The process has also been attempted in disks [98], although at very high PW_{50}/T the advantage from linearization is lost because the signal levels are so small that the head appears almost linear. In the credit card system direct linearization is possible, if difficult, due to the reasons outlined above. In a Bayesian detector it is only necessary to find the simpler $\beta(\cdot)$ rather than $\beta^{-1}(\cdot)$ (see later, section 5.5.4). There are, however, cruder, more practical methods available.

Soft decision direction

When using adaptive equalization, threshold detectors have exactly the wrong updating properties with equalized sample levels near to zero. The poor error signal is likely to have been caused by non-systematic problems in the read process (for instance a defect in the medium), and e^2 will be large. Unfortunately the values of h^{-1} are changed the most by large values of e. A squelch system is useful, and this is known as soft decision direction.

Such a method [74] can provide very considerable gains when used in a DFE system at low SNR when decoding errors are common. It is known that in any feedback detector, incorrect decisions can produce substantial error propagation [6, 58]. Another very similar mitigation method is to freeze the tap coefficients completely when the signal is poor [92].

DFE: decision feedback equalization

5.2.4 Blind adaptive equalization

$r(t)$ = read signal
$w(t)$ = written signal
$h(t)$ = channel response
$n(t)$ = noise signal

Suppose $r(t)$ is known, but $h(t)$, $w(t)$ and $n(t)$ are unknown. Is it still possible to find $h^{-1}(t)$ and equalize this channel? The answer, surprisingly, is yes, given time and some knowledge of $w(t)$. The technique is known as blind equalization, and it is gaining popularity because of its ready application to mobile telephone data channels.

DFE: decision feedback equalizer

One method already alluded to is the decision feedback equalizer (DFE). This is where it is assumed that the channel corruption, $h(t)$, is slight enough

that the unequalized channel produces correct output values more often than not. These decisions are fed back to a filter to produce a signal that cancels out the inter-symbol interference (ISI). The feedback signal can then be used as a training signal for adaptive equalization [16].

ISI: inter-symbol interference

Blind equalization schemes become most effective when combined with a detection mechanism.

5.2.5 What makes a 'good' channel compensator?

All channel compensation schemes must introduce extra noise when signal information has been lost by the channel. Not pushing the channel to the limit where it loses information is inefficient: all the extra information that $w(t)$ has over w_i is redundant. This implies that the best equalization scheme is the one that introduces the least noise, as this allows the highest *useful* information content. The equalizer that introduces the least noise is no equalizer at all (a matched filter reduces the apparent noise at the expense of bandwidth), so there would appear to be a huge advantage to be gained in matching the written signal to the channel, or to dispose of the equalizer altogether. This is discussed in section 5.3.

It must be noted that in the card reader context, the standard equalization schemes mentioned require a fixed channel sample rate that the card reader cannot provide. The credit card read apparatus used for this project does not have the capability of directly tracking the card's velocity, so neither a fixed equalizer nor an adaptive equalizer can be used directly in the system without modification, although it would be possible to digitally alter the sampling rate after the fact.

5.3 Pre-equalization and partial response

A number of different pre-equalization schemes exist to take advantage of any known qualities of the data channel. Suppose that all the pertinent properties of

h = channel impulse response

$h(t)$ are known, then the modulation code used on the channel can be tailored to meet its spectral requirements.

One set of pre-equalizer schemes, known as partial response classes, are in widespread use. For instance, partial response class 4 (PR4 is formed by writ-

D = delay operator

ing $w_i - w_{i-2} \equiv (1 - D^2)w_i$ to the channel, where D is the delay operator and subtraction in the binary code domain is the exclusive-or operation). This does nothing to the spectrum of the outgoing data stream, but if the channel is subsequently accurately bandlimited to a parabolic power spectrum, maximized at 0.25 of the bit-rate and zero at 0 Hz and 0.5 of the bit rate, a very simple detection method emerges. The read-back values, r_i, when sampled at the correct interval take on three distinct values: $+r_p$, 0 and $-r_p$. A channel that was not bandlimited would give back two distinct values, $+r_u$ and $-r_u$. A channel that does not have the required response would take on many values for r, the particular value being dependent on the data.

Another class, extended PR4 (EPR4), with system polynomial $(1 + D - D^2 - D^3)w_i$, gives five read-back values: $+4r_e$, $+2r_e$, 0, $-2r_e$ and $-4r_e$ with a longer coded impulse response. This allows a slightly higher bit density that PR4.

The primary advantage of these precoders is that their required spectra, shown in figure 5.15, are fairly close to the natural spectrum of a magnetic channel (see figure 5.6 for the spectrum of a credit card). This means that the equalizers employed in the read side alter the signal than would be required to equalize a pure binary code. This in turn means that noise amplification by the equalizer is very much reduced, and hence bit densities can be higher. EPR4 with maximum likelihood detection, known as EPRML, has been shown to have a density advantage over equalized peak detection (section 5.4.1) of at least 20% in a hard disk environment [85]. However, due to the unknown nature of the channel in a card reader system, such precoding is, unfortunately, not an advantage.

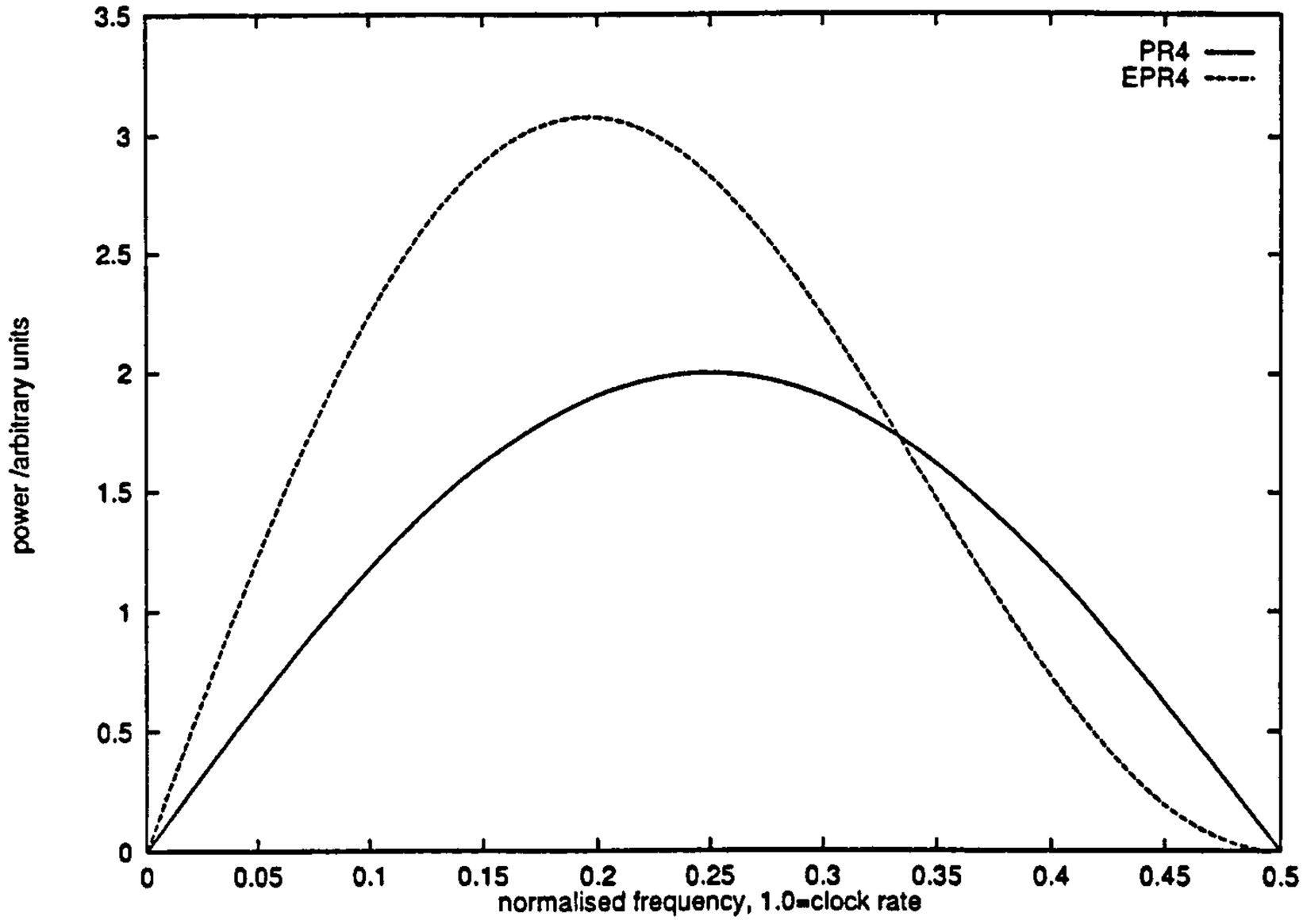


Figure 5.15: Required spectra for two partial response classes.

5.4 Common detection schemes

Detection is the process of turning an analogue signal, $r(t)$, back into digital data, $\hat{w}(t)$, that resembles the original data, $w(t)$.* The difference between detection and equalization is that a detector's output normally operates in the digital domain, and equalizer's output in the analogue, but the terms are often interchangeable. There are a wide range of detection methods, giving trade-offs between complexity and degree of correctable ISI. Even a perfect equalizer cannot reconstruct $w(t)$ exactly under all conditions, so the detection system has a significant role to play.

ISI: inter-symbol interference

The types of detectors can be classified into two broad classes. The dis-

*Please note the following slight change in notation for the rest of this section. Previously, $\hat{w}_i$ represented the *equalized* analogue channel whereas now it represents the digitally *decoded* channel estimate. In a perfect and lossless system the values are degenerate: $\hat{w}_{\text{old}} = \hat{w}_{\text{new}} = \hat{w}$. On the few occasions that $\hat{w}_{\text{old}}$ (estimated write signal from equalized read-back signal) is required in this changed context, the notation r' (equalized or modified read-back signal) shall be used.

crete detectors including peak and zero-crossing detection, and the maximum a posteriori* (MAP) detectors. The MAP detectors can be further sub-classified into separate-equalizer systems—including maximum likelihood, the Viterbi algorithm and the fixed-delay tree search (FDTS) where the equalization and detection are separate (and mostly independent) mechanisms—and combined-equalizer systems such as the Bayesian detector and decision feedback equalization (DFE). The FDTS algorithm looks briefly into the 'future', as does the Bayesian detector; all the other methods look only back at the 'past'.

MAP: maximum a posteriori

FDTS: fixed-delay tree search

DFE: decision feedback equalizer

5.4.1 Peak detection and zero-crossing detection

The simplest common detection method is called peak detection: it finds the peaks in $\frac{dr}{dt}(t)$. The extrema in the differentiated read-back signal normally occur at the zero-crossing points in $w(t)$. A simple analogue comparator circuit can locate these transitions and so $\pm w(t)$ can be reconstructed. A threshold-level gate is normally used to prevent spurious triggering on noise and to resolve the sign of $w(t)$.

r = read-back signal
w = written signal
$\hat{w}$ = write estimate
e = error signal

With equalization [90], the peak detector has proved itself to be extremely reliable and versatile in the general magnetic recording field. When there is significant ISI however, peak shift (the phenomenon of the peaks in $\frac{dr}{dt}(t)$ moving in time and decreasing in amplitude due to the linear superposition of signals) becomes a noise problem. If the channel is regular and consistent then some form of write-precompensation can be used to alleviate the ISI so that the filter-induced noise can be reduced.

5.4.2 The Viterbi algorithm

The Viterbi algorithm [31], when used to its fullest (infinte) extent on a perfectly linear and equalized channel, gives the lowest possible systematic error rate.

Very simply, the Viterbi algorithm is a method of estimating the minimum

*A posteriori means from effect to cause: i.e., from the effect of the written signal (the read-back signal) a MAP detector attempts to determine the written signal that caused it.

MLSE: minimum least squared error

least squared error (MLSE) condition. Defining $e_i = r_i - \hat{w}_i$, i.e., the error is the difference between the (equalized) read-back sample and the *decoded* data value, the MLSE condition is the value of $\hat{w}$ that minimizes $\sum_i e_i^2$. The raw MLSE calculation requires that every value of r_i is known, and that every combination of $\hat{w}_i$ is tried against r. As the number of samples is likely to be very large, and the detection procedure takes time of order $\mathcal{O}(2^{\text{number of samples}})$, the full calculation is practically out of the question.

The Viterbi algorithm is a method of getting very close to the MLSE condition with a computation time $\mathcal{O}$(number of samples) and arbitrary limits on memory requirements (hence decoder complexity). For example, on continuous satellite data channel systems a few hundred past sample paths are recorded. (In the case of radio channels, phase and amplitude modulation tends to be used instead of binary signaling; however the extension of the algorithm is straightforward.)

It has recently been shown that it is possible to get within 10% of the Shannon limit with Viterbi detection and an iterative recursive punctured systematic coding scheme. The so-called 'turbo-codes' (because once correct bits are decoded, this increases the likelihood of future bits being decoded correctly: a turbo-charger effect) [8] are emerging in the research field.

5.4.3 Trellis coding and detection

The MLSE condition can easily make single bit errors if there is a short noise burst at that point. Adding check bits to data blocks is one method of detecting (and perhaps correcting) these errors. Another method is a convolutional coding scheme. This relies on the coder inserting redundant bits to separate the Hamming distance between possible paths. When drawn graphically, these paths look like trellis, hence the name. The code's error correction ability relies on all paths converging to one (or more) state(s) after a predetermined amount of time. Then the Viterbi detector looks at the whole section of trellis, decides on the most likely path of the received signal, and steps forward in time to the

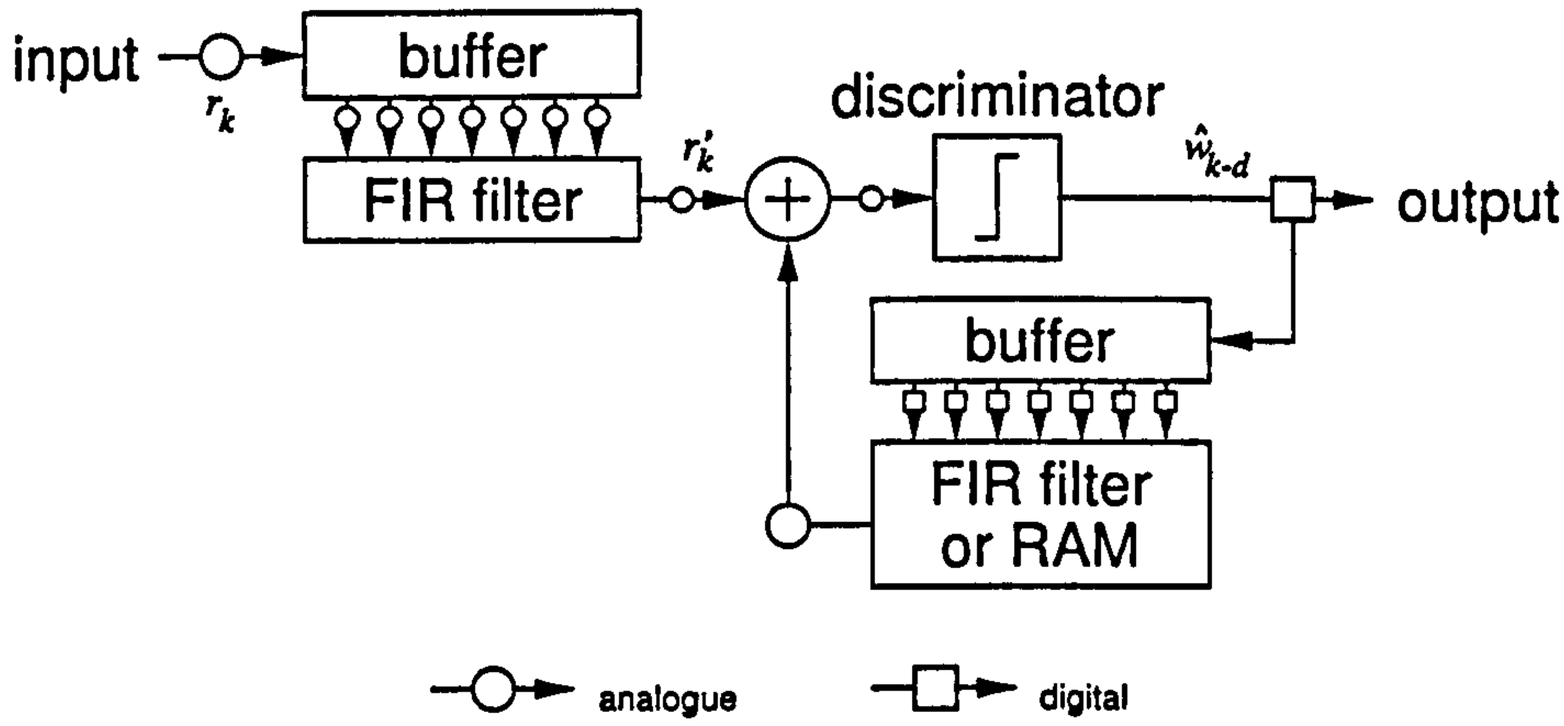


Figure 5.16: Structure of conventional decision feedback equalizer.

next trellis section. If the paths through the section of trellis are well separated, the Viterbi detector will provide quite substantial error correction power.

Convolutional trellis coding is often used as a first stage wrapper, removing or locating single bit errors before passing the data up to a block error detection and correction stage. It can also prevent extensive code error propagation, mitigating some of the effects of a timing error.

Trellis coding is just appearing in commercial hard disk products, where it is claimed that it has 1.5 dB noise margin advantage over EPRML detection [29].

EPRML: extended partial response and maximum likelihood

5.4.4 Decision feedback equalization

ISI: inter-symbol interference

In the ISI channel the value of one bit affects adjacent samples. If the approximate channel response is known, then the trailing ISI portion can be canceled by using a decision feedback equalizer (DFE). The structure is shown in figure 5.16: once the data values are decoded, their ISI contribution to future samples is subtracted from the incoming signal. Note that the channel is pre-filtered before passing into the DFE. This is often a simple noise whitening filter but it can also attempt to remove the leading part of the impulse response.

Although the channel response must be known, there are straightforward procedures to modify the feedback filter so that it can adapt, blindly, to the

channel [82].

NLTS: non-linear transition shift

In very high density magnetic recording, non-linear transition shift (NLTS) can upset the linear portion of the DFE system. NLTS is the phenomenon of magnetic boundary shifting when the content of one bit cell affects the rise-time of the write magnetic field in a neighbouring bit cell. This effect is mostly backwards in time, i.e., written bits affect future bits to be written. NLTS is also affected slightly by the old data that is being erased, but since the erasure takes place at the leading edge of the write head and storage occurs at the trailing edge of the write head this effect is small. The backwards direction makes NLTS perfect for being corrected by DFE, and a temporal non-linear channel filter can be made simply by replacing the feedback FIR filter with a block of random access memory (RAM). The RAM-DFE [19, 30] gives a significant improvement over normal DFE in high density situations, and the RAM values can be updated blindly with data taken from the channel.

The DFE can also be used where channel precoding has been used, for example EDFE [100] uses a band spectrum that gives an infinite tail on the impulse response, but the consequent deliberate 'ISI' is completely corrected by the DFE.

EDFE: enhanced decision feedback equalizer

5.5 Bayesian detection

w_i = written signal
$\hat{w}_i$ = decoded signal
r = read signal

A linear FIR equalizer estimates the original data, $\hat{w}_{k-d}$, from linear superpositions of samples in a buffer, r, of length n, $r_k = (r_{k-n+1}, r_{k-n+2}, r_{k-n+3}, \ldots, r_k)$. The current time index is subscripted k, but there is an intentional delay, d (for a symmetric filter $d = n/2$)*, to allow 'future' values of r to improve the accuracy of $\hat{w}$. The linear equalizer produces an analogue result from an analogue channel, but it is known that the original signal, w, was digital.

k = current time
d = delay
n = sample buffer length

h = channel response

Given r_k, it is possible to make better decisions than simply $h^{-1} \cdot r_k$. In fact, the discrete probabilities that $w_{k-d} = +1$ and $w_{k-d} = -1$ can be calculated

*This is a new use of d, and it is not the same d as in (d, k) expressions, nor is it the d operator used in differentiation.

from r_k. The proof of the calculation [2] involves Bayes' theorem, hence these detectors are known as Bayesian detectors. The calculation is outlined below.

Bayes' theorem states that the conditional probability of an event, α, given that another event, β, has occurred can be computed.

$$\text{pr}(\alpha|\beta) = \frac{\text{pr}(\alpha,\beta)}{\text{pr}(\beta)} = \frac{\text{pr}(\beta|\alpha)\,\text{pr}(\alpha)}{\text{pr}(\beta)} \tag{5.21}$$

This can be extended trivially to cover multiple events.

$$\text{pr}(\alpha|\beta,\gamma)\,\text{pr}(\beta|\gamma) = \text{pr}(\alpha,\beta|\gamma) = \text{pr}(\beta|\alpha,\gamma)\,\text{pr}(\alpha|\gamma) \tag{5.22}$$

For independent events, the simple probability rules apply.

$$\text{pr}(w_i, w_j|\gamma) = \text{pr}(w_i|\gamma)\,\text{pr}(w_j|\gamma) \qquad i \neq j \tag{5.23}$$

$$\text{pr}(w_i|\gamma) = \sum_{\text{all } s} \text{pr}(w_i, w_j = s|\gamma) \qquad i \neq j \tag{5.24}$$

The objective is to derive

$$\zeta_k(s_d) = \text{pr}(w_{k-d} = s_d|r_k, \hat{w}_{k-d-1}, \hat{w}_{k-d-2}, \ldots, \hat{w}_{k-m+1}); \tag{5.25}$$

s = test variable

m − d = past decision buffer size

i.e., the probability that the $(k-d)^{\text{th}}$ written symbol had the value s_d. Notice that in addition to knowing the read-back values, r_k, the $m-d$ past decisions are also known, $\hat{w}_{k-d-1} \ldots \hat{w}_{k-m+1}$ and they will be allowed into the calculation. This feedback will reduce the computation considerably. Also note that although most subscripts count up in time $(w_0, w_1, w_2, \cdots, w_k)$ the s variables here are indexed by delay, and hence appear to count down in time $(s_m, s_{m-1}, s_{m-2}, \cdots, s_0)$.

To proceed, the tentative write events, w, are expanded one level into the future.

$$\zeta_k(s_d) = \sum_{s_{d-1}} \text{pr}(w_{k-d} = s_d, w_{k-d+1} = s_{d-1}|r_k, \hat{w}_{k-m+1}, \ldots, \hat{w}_{k-d-1}) \tag{5.26}$$

And, indeed, it is useful to expand the sum very much further (to d' beyond k).

$$\begin{aligned}
\zeta_k(s_d) &= \sum_{s_{d-1}} \cdots \sum_{s_{-d'}} \mathrm{pr}(w_{k-d} = s_d, \ldots, w_{k+d'} = s_{-d'} | \boldsymbol{r}_k, \hat{w}_{k-m+1}, \ldots) \\
&= \left(\mathrm{pr}(\hat{w}_{k-m+1} = s_{m-1}, \ldots, \hat{w}_{k-d-1} = s_{d+1} | \boldsymbol{r}_k) \right)^{-1} \cdot \\
&\quad \sum_{s_{d-1}} \cdots \sum_{s_{-d'}} \mathrm{pr}(\hat{w}_{k-m+1} = s_{m-1}, \ldots, \hat{w}_{k-d-1} = s_{d+1}, \\
&\quad w_{k-d} = s_d, w_{k-d+1} = s_{d-1}, \ldots, w_{k+d'} = s_{-d'} | \boldsymbol{r}_k) \qquad (5.27)
\end{aligned}$$

(The equality comes straight from Bayes' theorem, equation (5.22), where a new set of events has now been forced into the expression.) Note that the first factor in (5.27) is constant with regard to s_d, and will be removed from the active calculation. In the case that the feedback is correct, i.e., $\hat{w}_{k-i} = w_{k-i} = s_i$ for $i > d$, the following results are evident.

$$\begin{aligned}
\zeta_k(s_d) &= \phi_k \sum_{s_{d-1}} \cdots \sum_{s_{-d'}} \mathrm{pr}(w_{k-m+1} = s_{m-1}, \cdots, w_{k+d'} = s_{-d'} | \boldsymbol{r}_k) \qquad (5.28) \\
&= \phi_k \sum_{s_{d-1}} \cdots \sum_{s_{-d'}} \mathrm{pr}(\boldsymbol{r}_k | w_{k-m+1} = s_{m-1}, \ldots, w_{k+d'} = s_{-d'}) \cdot \\
&\quad \frac{1}{\mathrm{pr}(\boldsymbol{r}_k)} \mathrm{pr}(w_{k-m+1} = s_{m-1}, \ldots, w_{k+d'} = s_{-d'}) \qquad (5.29) \\
&= \phi'_k \sum_{s_{d-1}} \cdots \sum_{s_{-d'}} \eta_k(s_{m-1}, \ldots, s_{-d'}) \eta'_k(s_{m-1}, \ldots, s_{-d'}) \qquad (5.30) \\
&= \phi'_k \sum_{i=0}^{2^{d+d'}-1} \eta_k^{(i)} \eta_k'^{(i)} \qquad (5.31)
\end{aligned}$$

Here, ϕ_k and ϕ'_k are simply collections of the normalizing factors that are constant over the sum of interest:

$$(\phi'_k)^{-1} = \mathrm{pr}(\boldsymbol{r}_k) \, \mathrm{pr}(\hat{w}_{k-m+1} = s_{m-1}, \ldots, \hat{w}_{k-d-1} = s_{d+1} | \boldsymbol{r}_k) \qquad (5.32)$$

Equation (5.29) is again from Bayes' theorem (5.21), but (5.30) and (5.31) are notational shorthand. The new label, $\eta_k(\cdots)$, is a calculable probability that will be the active computation and $\eta'_k(\cdots)$ is the a priori code sequence probability; in this context it acts as a dynamic normalizing factor. (It would be undesirable

to include probabilities from an impossible code sequence in the calculation.)

$$\eta_k(s_{m-1},\ldots,s_{-d'}) = \mathrm{pr}(\boldsymbol{r}_k | w_{k-m+1} = s_{m-1},\ldots,w_{k+d'} = s_{-d'}) \tag{5.33}$$

$$\eta'_k(s_{m-1},\ldots,s_{-d'}) = \mathrm{pr}(w_{k-m+1} = s_{m-1},\ldots,w_{k+d'} = s_{-d'}) \tag{5.34}$$

The parenthesized superscript in (5.31) is another useful shorthand to combine the values of s_i into a single index. For binary signaling ($s_i \in \{+1,-1\}$) this index is simply the binary representation of $s_{m-1},\ldots,s_{-d'}$.

$$\eta_k(s_{m-1},\cdots,s_{-d'}) \equiv \eta_k^{\left(\sum_{j=-d'}^{m-1}(s_j+1)\cdot 2^{(d'+j-1)}\right)} \tag{5.35}$$

In summary, the expression for the probability that $w_{k-d} = s_d$ given in (5.25) using the set of samples in $\boldsymbol{r}_k$ and past decisions, $\hat{w}_i$ has been reduced to a summation of calculable probabilities. By comparing $\zeta_k(+1)$ with $\zeta_k(-1)$, the maximum a posteriori value of w_{k-d} is determined: the MAP value being the more likely.

MAP: maximum a posteriori

There is also the question of how large n (the size of the sample buffer in $\boldsymbol{r}_k$) and m (the number of feedback decisions less the delay, d: it also temporally binds $\hat{w}_{k-i}$ through $m > i > d$) should be. This depends to some extent on the range of the suspected ISI (figure 5.17): if the length of the channel impulse response is l samples then sample r_k will be affected by written bits w_{k-l+1} to w_{k+l-1}. Thus the probability calculation (5.33) dictates that $l-1$ future and past bits are required ($d' = l-1$) and $m = n+l-1$. The value of n and the shape of the impulse response will determine the value of d. For instance if the response is symmetrical, then it appears sensible to make $d \approx n/2$. The value of n affects the error rate of the detector: higher n gives lower error rates.

However, this is not the only factor playing a part in determining the sizes of the filter elements. The computation time is mostly dependent on the look-ahead terms in equation (5.31): $\mathcal{O}(2^{d+d'})$. There is therefore great incentive to reduce the values of both d and d': the decision feedback part is computationally inexpensive, so d will probably be chosen to be smaller than $n-d$, even though

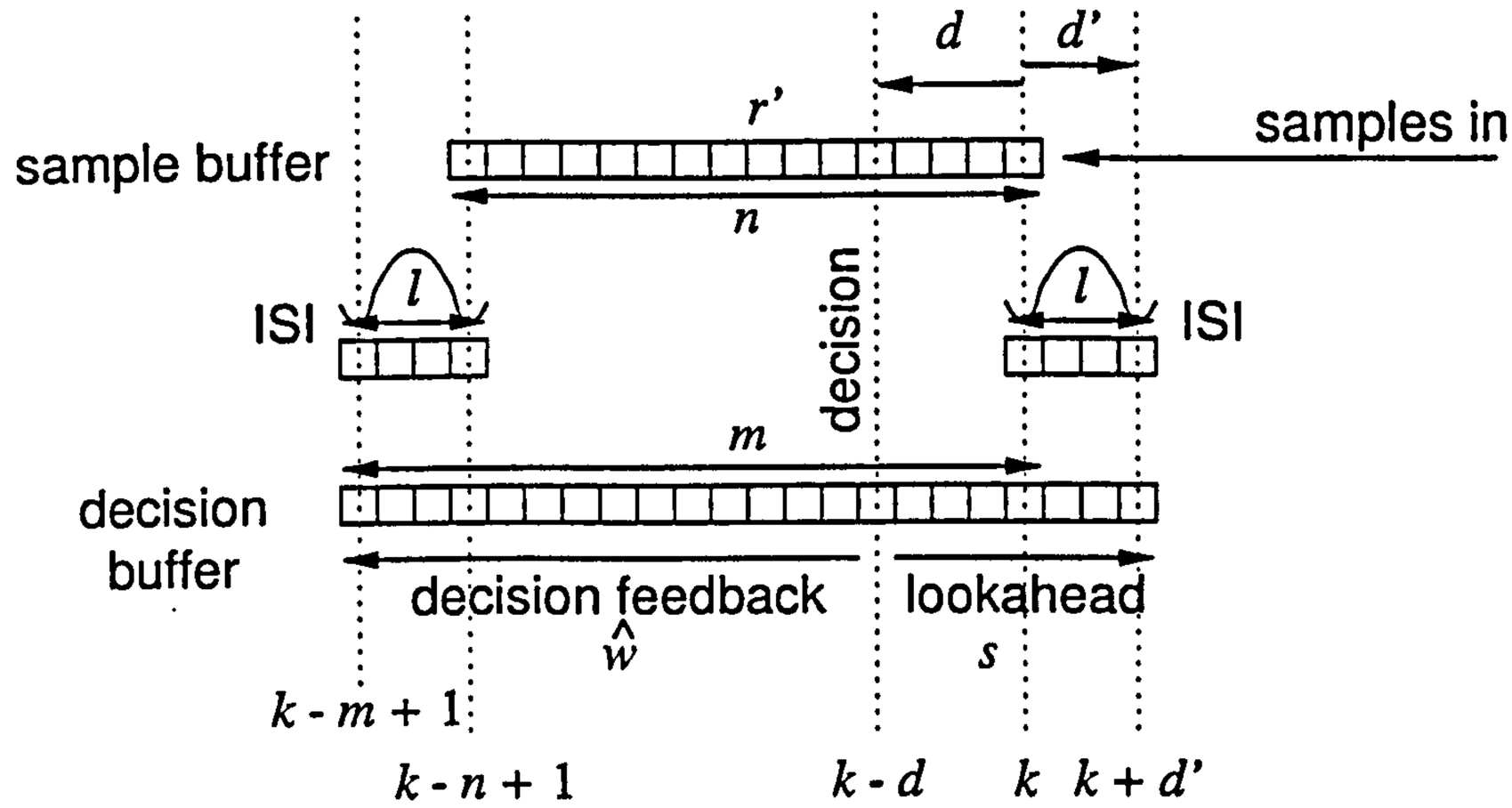


Figure 5.17: Time-line for Bayesian probability calculations.

doing so will increase the likely bit error rate. Additionally the timing recovery mechanism may rely on a minimum value of n.

It finally remains to calculate $\eta_k^{(i)}$. A simple model is to consider the difference between the *actual* sample value and the *theoretical* sample value then ascribe the difference between the two to noise. This requires a model of the channel to generate the theoretical sample values, $\hat{r}_k^{(i)}$, from $\hat{w}_k$ and s_i. (The theoretical sample value is the value that would be obtained if the channel model were totally correct. The model is usually close, but it is an estimate of the real channel at best. The worth of the theoretical sample value is that it is noiseless.)

ϵ = *noise pdf*

If the probability density function (pdf) of the noise is $\epsilon(r, \hat{r})$ and the noise is uncorrelated (hence r_i is uncorrelated with r_j for $i \neq j$) then $\eta_k^{(i)}$ can be calculated thusly.

$$\eta_k^{(i)} = \prod_{j=0}^{n-1} \epsilon(r_{k-j}, \hat{r}_{k-j}^{(i)}) \tag{5.36}$$

The basic structure of this form of transverse Bayesian equalizer is shown in figure 5.18. The $\eta_k'^{(i)}$ calculation is implicitly performed by a sequence generator that is used to make valid data sequences from the $2^{d+d'}$ possible uncoded combinations. From this $\hat{r}_k^{(i)}$ can be calculated and these values are compared with

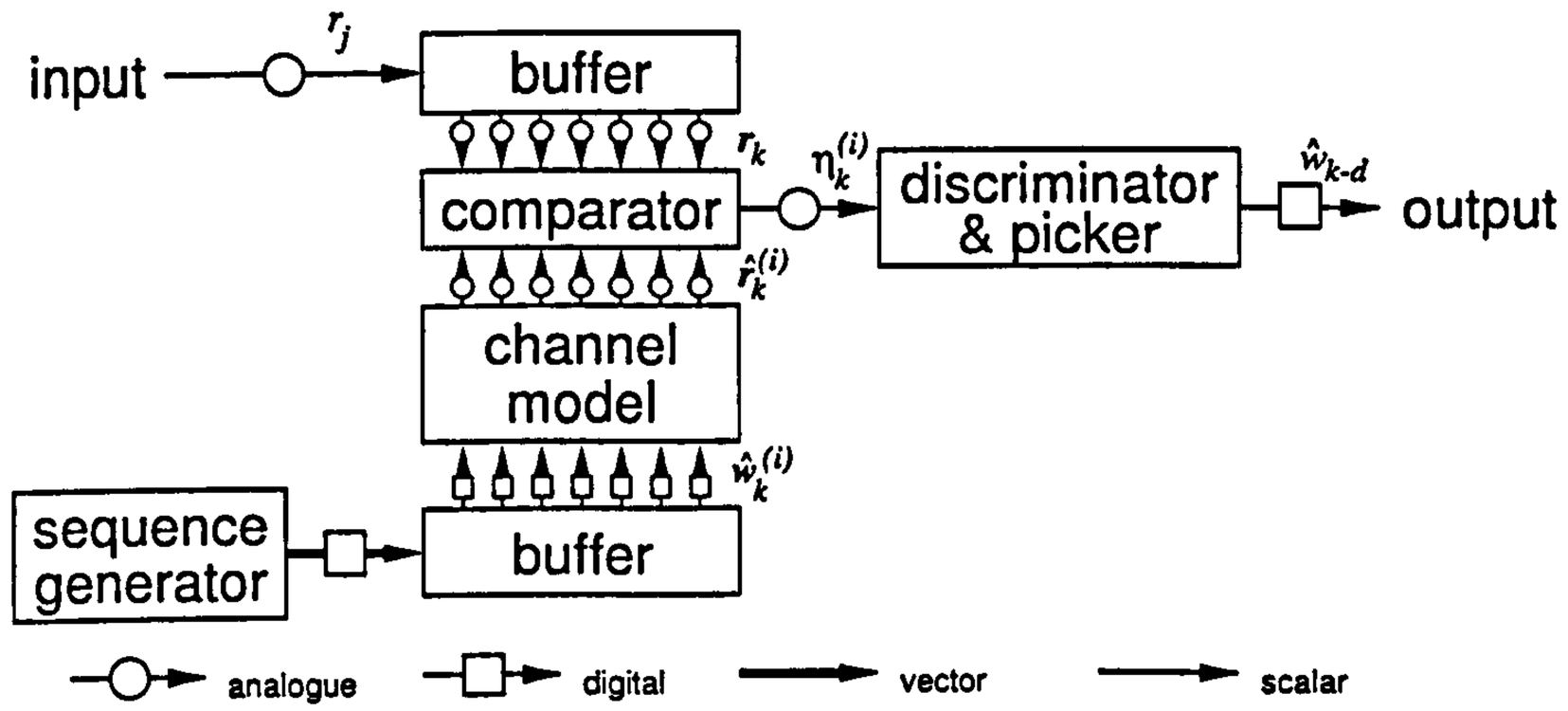


Figure 5.18: Structure of transversal Bayesian equalizer (no feedback).

r_k. The comparator passes $\eta_k^{(i)}$ to a 'picker' which sums $\eta_k^{(i)}$ to produce ζ_k. Depending on the values of ζ_k, a +1 or −1 is assigned to w_{k-d}. Then k increments as a new sample is read, and the process restarts. Since ϕ'_k is constant for fixed k, its value need not be calculated at all.

Despite its apparent complexity, the final calculations are not overly involved. Schemes similar to this have been showing promise in mobile radio systems [16] although their general use is usually supplanted by simpler detection systems.

5.5.1 Fixed-delay tree search

The first of the Bayesian detectors commercially evaluated in magnetic channels was the fixed delay tree search [39, 70]. This is a simplification of the Bayesian symbol detector described previously. The simplifications are designed to increase the speed of the equalizer and are outlined below.

- The channel is assumed to be linear and equalized. I.e., $\hat{r}_k = \hat{w}_k$.

iid: independent identically distributed

- The noise element is considered to be uncorrelated (iid) Gaussian white noise: $\epsilon(r_k, \hat{r}_k) \propto \exp(-\sigma^{-2}(r_k - \hat{r}_k)^2)$. This allows calculation of (5.36) in the log domain, i.e., (disregarding constants) $-\log \eta_k^{(i)} = \sum_j (r_{k-j} - \hat{r}_{k-j}^{(i)})^2$.

DFE: decision feedback equalization

- Previously determined values for $\hat{w}_{k-d}$ are fed back in the analogue domain with a DFE-like mechanism. This pairing is known as FDTS/DF and gives $m = n = d$.

- Normally $d' = 0$ and $d = 1$ or 2.

- The combinations of $s_{d-1}, \ldots, s_0$ are considered a tree of equiprobable binary branches.

- Since a single wrong test bit in the branch being searched rapidly makes $\eta_k^{(i)}$ very small, $\zeta_k(\cdot)$ is taken simply as the maximum value of $\eta_k^{(i)}$. This allows a Viterbi-like algorithm to be used to calculate the branch log-likelihoods thus substantially cutting the number of operations required.

These features are summarized in figures 5.19 and 5.20. Published simulations [70] show the FDTS/DF gives a 6–7 dB signal to noise ratio advantage over peak detection. It has been shown [55] that the full tree search is not necessary for certain subsets of codes and that a simpler DFE structure can be used instead. The DFE output is a multi-level signal that, when filtered, is almost equivalent to the FDTS. This approach, called MDFE, has an order of magnitude lower bit error rate at high signal to noise levels [61].

MDFE: multi-level decision feedback equalization

5.5.2 The blind adaptive Bayesian filter

Consider that the form of the *forward* channel is approximated to a transversal filter in $\hat{h}$. Also consider that the noise is iid white Gaussian. The following expressions are now evident.

$$\hat{r}_k^{(i)} = \hat{h} \cdot s^{(i)}(k) \tag{5.37}$$

$$\epsilon(r_k, \hat{r}_k) = (\text{constant}) \exp\left(-\sigma^{-2}(r_k - \hat{r}_k)^2\right) \tag{5.38}$$

In the above, $s_k^{(i)} = (s_{m-1}, \ldots, s_0)^T$ is the vector of tentative symbols currently being tested against r_k (refer back to equation (5.25) for further information) and σ is related to the standard deviation of the noise.

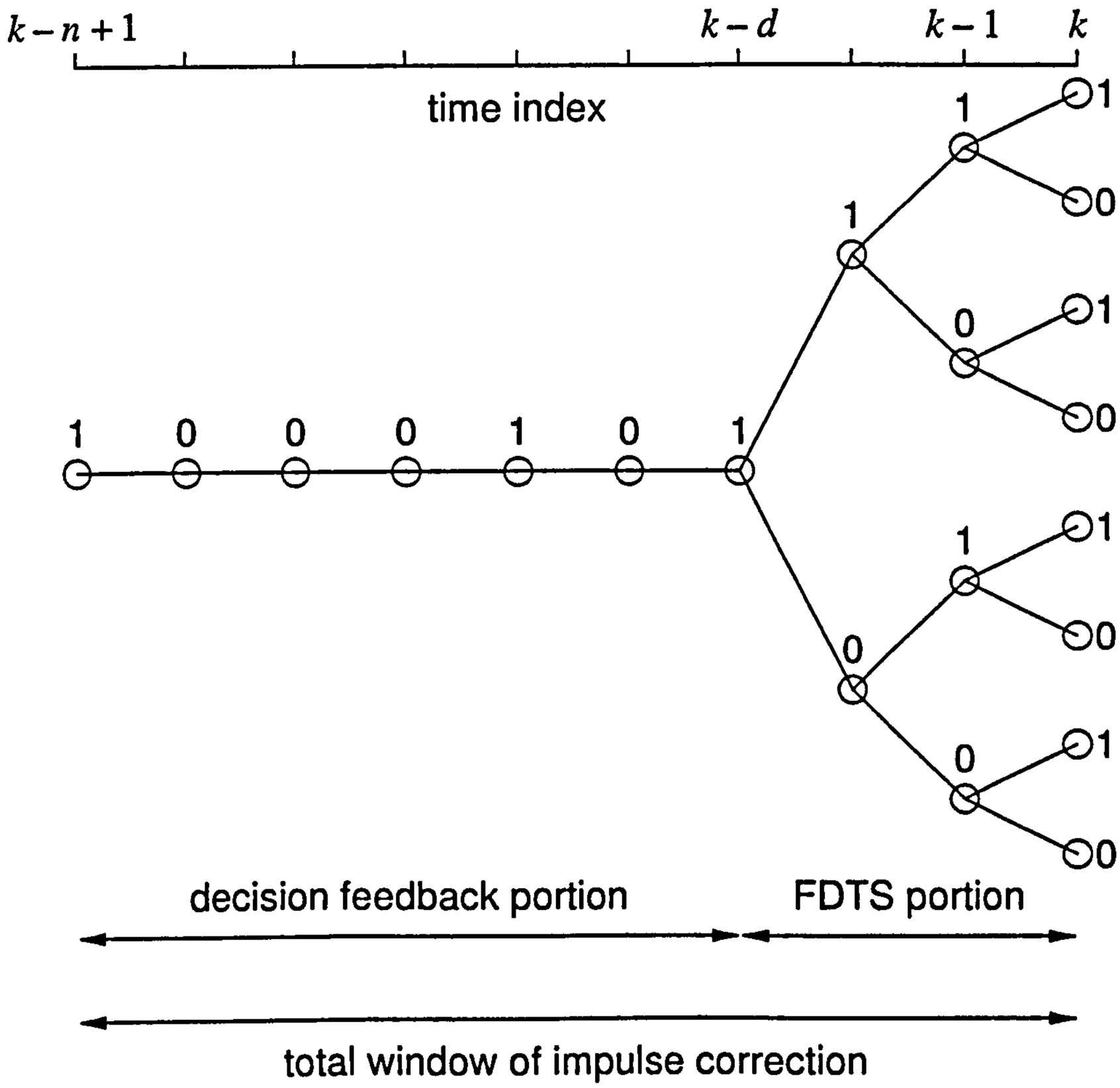


Figure 5.19: Principle of the tree search with decision feedback.

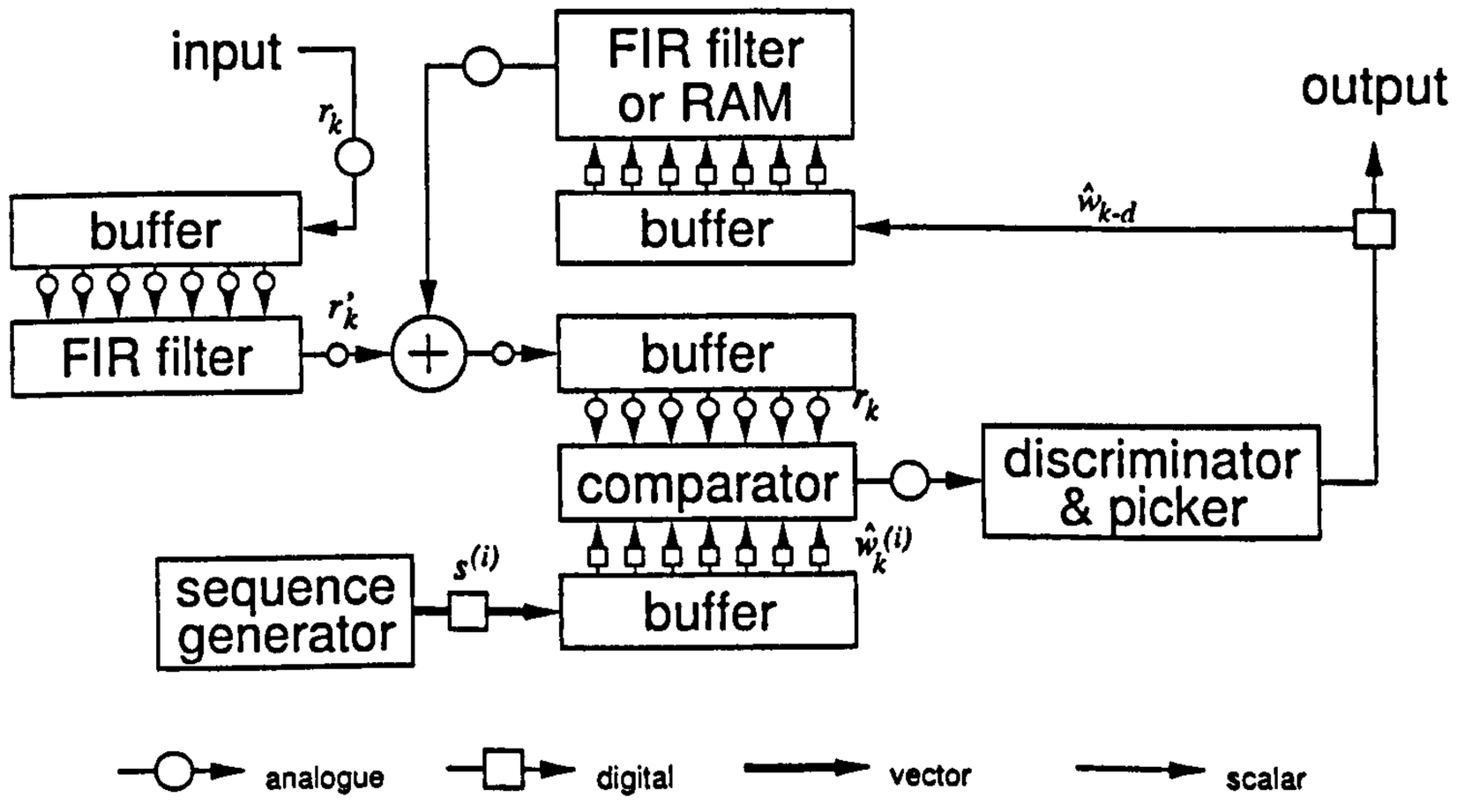


Figure 5.20: Structure of fixed delay tree search with decision feedback detector.

LMS: least mean squares

Given these assumptions, it is possible to update $\hat{h}$ in a blind channel situation: both the Kalman filter and the LMS adaptive filter can be used [43, 44]. LMS is probably more common [10, 16] but a direct channel estimator is also possible [35]. The LMS algorithm is simple and reasonably stable and was described in section 5.2.2.

Since (after the calculation of $\hat{w}_{k-d}$) there also is an error signal available, the LMS update vector is almost trivial to compute: compare equation (5.18) with the following.

$$\hat{h}_j \longleftarrow \hat{h}_j - \mu \hat{w}_{k-d-1-j}(r_{k-d} - \hat{r}_{k-d}) \quad \text{for} \quad 0 \leq j < m-d-2 \tag{5.39}$$

Notice that the range of j has been limited in this case to past (firm) decisions, rather than allowing the tentative decisions into the calculation.

5.5.3 Variable speed Bayesian detection

The credit card channel model used in this work is schematically depicted in figure 5.21. There is a substantial variable speed element and the sampling clock is not synchronized in any way with the speed of the channel. It is imperative therefore that the signal is oversampled, but for any detector to work, the channel clock must be recovered. Although there has been some activity in this area of asynchronous sampling for power saving in digital phones (utilizing Farrow structure digital interpolation filters [34, 28]) this narrow-band phase correction is not sufficient for the range of velocity present in the credit card reader. Instead the decision was taken to use fundamental sinc-based resampling methods on a buffer of asynchronously sampled data.

Consider the free running samples that are taken at sampling interval T'. It is desired to move samples from the T' buffer $r'_0, \ldots, r'_{k'}$ at offset t' into a buffer at the symbol rate T, $r_0, \ldots, r_k$. Assuming that T' will oversample the Nyquist

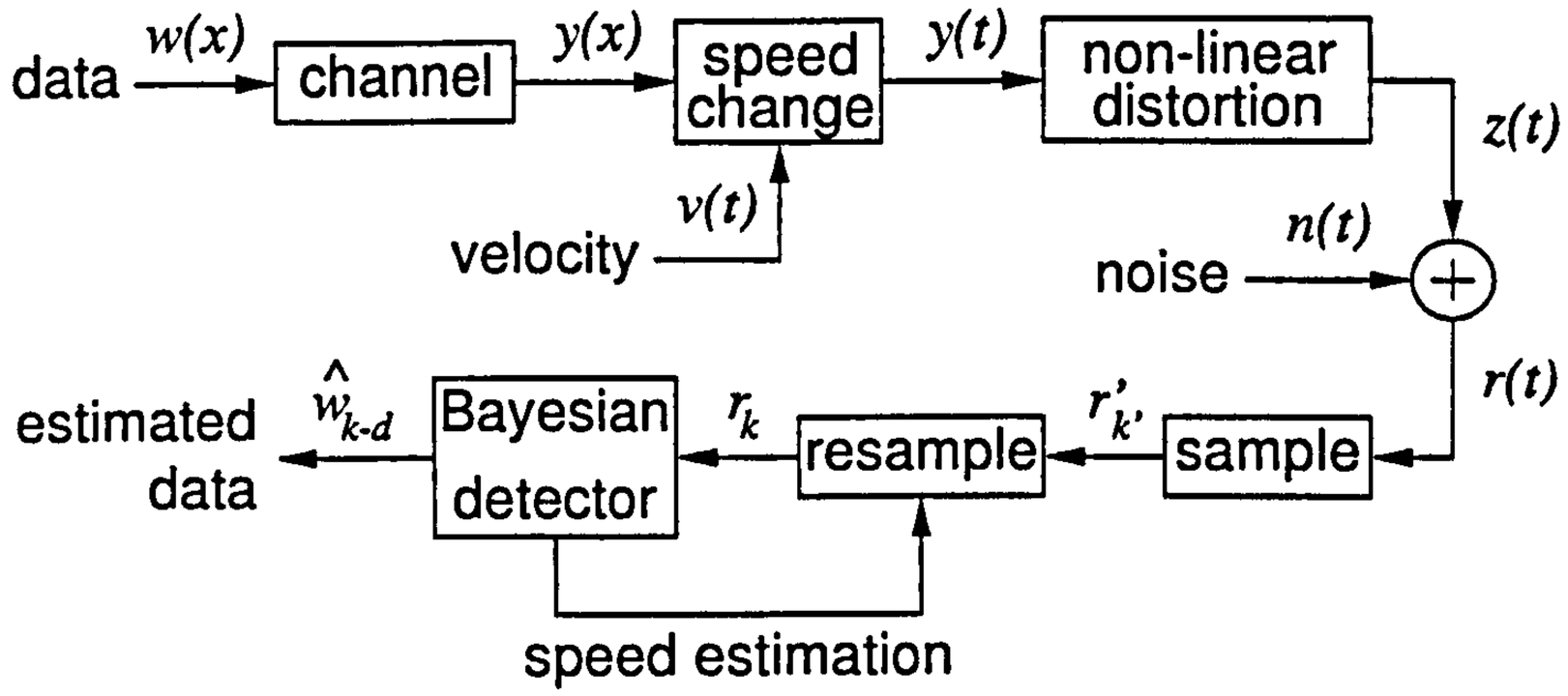


Figure 5.21: Block channel model for variable speed Bayesian equalizer.

frequency in $r(t)$, r_k can be derived as follows.

$$r_k = \sum_{k'=-\infty}^{\infty} \text{sinc}(t' + k'T' - kT) r'_{k'} \tag{5.40}$$

The $\text{sinc}(x) = \frac{\sin(x)}{x}$ function can be truncated in time with a windowing function [80] so that the summation is limited to a more conveniently computable boundary (say 11 samples) with very little apparent loss of accuracy.

It is now evident that there is a 'snatch' of signal available—$(r_{k-n+1}, \ldots, r_k)$ as derived from the asynchronous samples, $r'_{k'}$ in (5.40) above—and also from the Bayesian detector structure described previously—a 'snatch' of theoretical signal $(\hat{r}_{k-n+1}, \ldots, \hat{r}_k)$. If the sampling rates of these two buffers are not too dissimilar (r_k is derived from an estimate of the data clock—not the actual data clock—while $\hat{r}_k$ *is* derived from the data clock) then a more accurate value of T' can be calculated from the two snatches: this process was described in section 4.2, page 81.

The structure of this new detector is shown in figure 5.22. The process adds a resampler and a velocity estimator to the Bayesian feedback detector. Although a commercial system would most likely implement an automatic gain control (AGC) feature too, this was not included: the overall gain control was

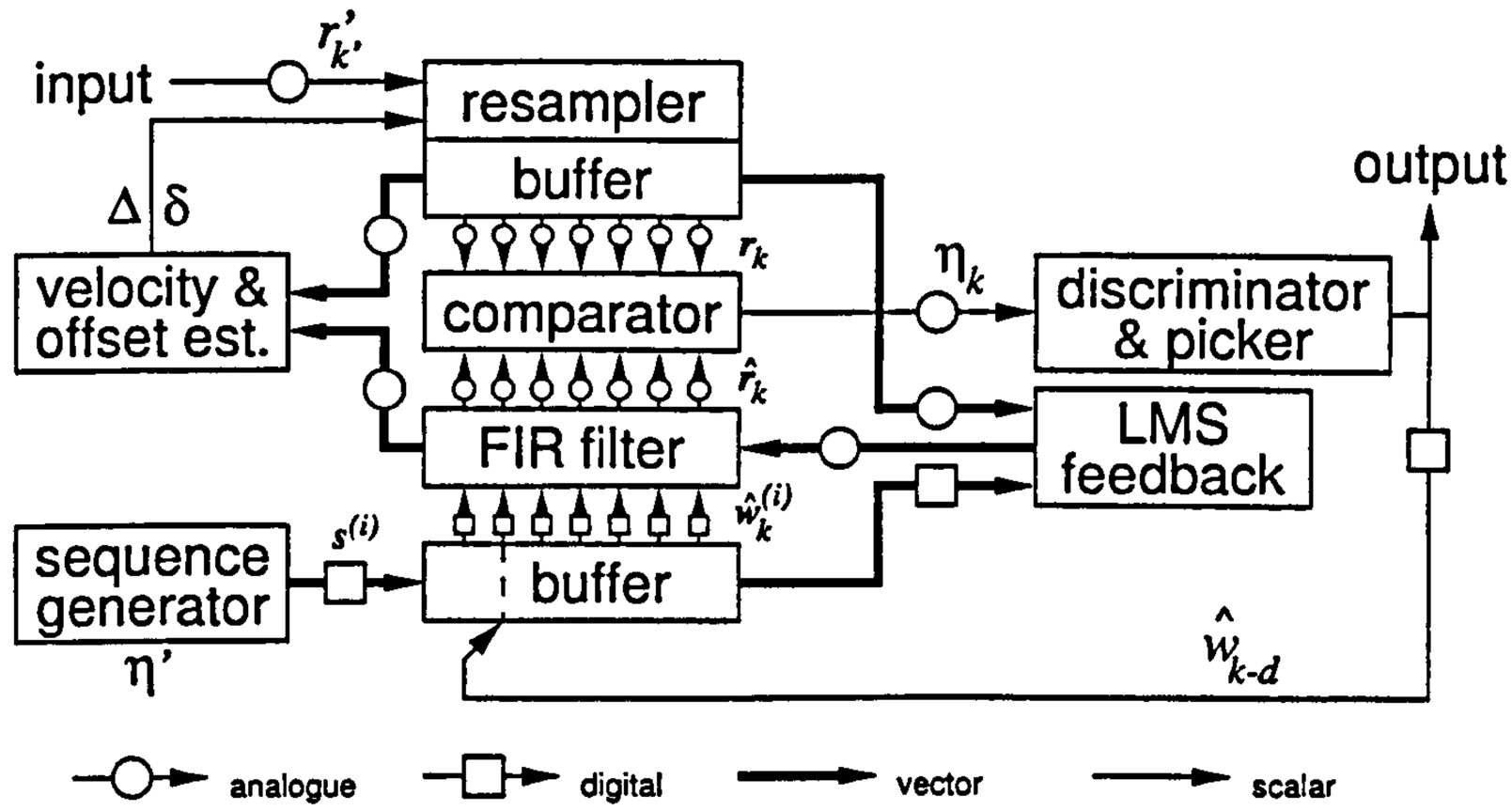


Figure 5.22: Structure of variable speed Bayesian equalizer.

set statically by hand and small dynamic alterations were covered by the LMS filter feedback. This arrangement worked because (with an MR detector) the only amplitude changes were due to dropouts where the head-medium spacing changed; so the channel response also changed. An AGC may respond too rapidly to channel changes to work well at correcting this type of fault, and it would introduce yet another feedback path (albeit within an independent contained loop).

AGC: automatic gain control

5.5.4 Non-linear distortion in a Bayesian detector

There remains the question governing the effects that the non-linear distortion (shown as $z(t) = \beta(y(t))$ in figure 5.21) has on the system. It has already been shown (section 4.2.3, page 85) that correcting this distortion, at least by truncation, reduces the effectiveness (although improving the consistency) of the snatch method of timing recovery. However, linear systems do not take well to non-linearities in general: two studies have shown that distortion as small at 5% can cause an order of magnitude increase in the bit error rate of PRML detectors [102, 107]. Newly published approaches have suggested relineariza-

β = distorting function

PRML: partial response, maximum likelihood

tion [98] and neural network based equalizers [72, 103] as methods to tackle this problem. The latter approach has achieved superb results, with the proviso of a potentially lengthy training sequence. In the variable speed Bayesian system presented, the FIR filter and discriminator will be affected by distortion; the consequences of this will be examined shortly.

FIR: finite impulse response

The Bayesian detector per se does not *require* a linear input. It bases its decisions on the calculation of probability that one signal was caused by another. That probability calculation is *refined* by possession of knowledge of the channel: the more information on the distortion function, noise characteristics, non-linearities and so forth, the more technically accurate the decisions will be. It should be pointed out once again, that this is knowledge of the forward channel, not the inverse channel. This is a significant advantage, because the forward channel is, at least in principle, a measurable commodity, although its measurement may not be trivial in a blind channel.

Given this basis, is it worth examining the channel in minute detail to retrieve extra bits? In a long duration channel, where the statistics will be meaningful over a considerable length of time, it may well be worth the effort. In a shorter burst channel, where the set of measurable statistics is not much smaller than the set of bits to be decoded, this potential advantage evaporates. The blind channel algorithms would adapt, but are compelled to do so in such a short amount of time that they may determine the data bits themselves as mere channel parameters.

The decision was taken not to attempt non-linear compensation, but to reduce the effects of non-linearities on the feedback filter and η_{k-d} calculation by the following qualitative argument. What is the likelihood that a large, positive read-back signal was caused by noise alone? While it is not possible to make comprehensive measurements of all credit cards, the answer remains that the likelihood is vanishingly small.

MR: magneto-resistance

When MR flip-back (examples are shown in figures 1.9 (page 17) and 5.43) occurs, the signal at the flip-back is still large and positive. It seems probable, on the basis that noise on this scale is unlikely, that the signal should be large and

positive, rather than a superposition with a small negative signal. This is reinforced by the knowledge that the (d, k) code used is incapable of writing such a small transition (especially when the d parameter is non-zero). The calculation under these circumstances should, therefore, be forced.

s = test signal
$\hat{w}$ = past decisions

The following threshold truncation algorithm was used. When both the read-back signal, r_k, and the estimated read-back signal, $\hat{r}_k$ derived from s and $\hat{w}$, were in agreement that the signal was large and positive or large and negative, the probability of one causing the other is unity. Using white Gaussian noise as the background, this equates to the following expression.

$$\epsilon(r_k, \hat{r}_k) \propto \exp(-\sigma^{-2} \cdot g^2(r_k, \hat{r}_k)) \tag{5.41}$$

where with upper and lower thresholds,r_u and r_l, of r are used thus

$$g(a,b) = \begin{cases} 0 & \text{if } a > +r_u \text{ and } b > +r_u \\ 0 & \text{if } a < -r_l \text{ and } b < -r_l \\ (a-b) & \text{otherwise} \end{cases} . \tag{5.42}$$

The function $g(\cdot)$ is acting like a truncated subtraction. This is not a perfect solution (for one thing it has a tenuous basis in reality and for another the integral of the error pdf, ϵ, is not proper) but it does have some arguments in its favour. Firstly it is very simple to compute. Secondly it is robust, in the sense that it cannot fail or become unstable. The threshold levels need to be set up on the basis of the head and amplifier only: it is independent of the card being read. Thirdly, it appears to work (see the next section). Fourthly and finally, when $g(\cdot)$ effectively sets r_i to $\hat{r}_i$, it stops the LMS adaptive filter feedback from affecting the FIR filter. It does this in exactly the places where the FIR filter fails *because* of the non-linearities. This double gain was felt worthwhile enough to discontinue evaluation of more complicated and specific compensation schemes on the credit card read system.

LMS: least mean squares

5.6 Results with the velocity-tracking detector

SNR: signal to noise ratio

It is no trivial task to measure and classify the effectiveness of a system that was designed to cope with unclassifiable systems. The standard measure of a detection system is to plot the bit error rate against SNR or bit density. In the credit card system, the SNR is not measurable or controllable to a consistent degree. To artificially induce noise in the process is not realistic, because the real system errors are not noise induced. To use raw bit density as a measure is also unrealistic, because the effective user bit density is continually changing, and the value of the system lies in its ability to adapt. A full development study would calculate the *system* error rate against card capacity in field conditions, but that is beyond the scope of this study.

The following sections therefore present spot results from real cards and simulated signals. It is hoped that this method is more representative of the research than the presentation common in the literature, and conveys a more realistic picture than figures alone can display.

5.6.1 Initial parameters

For the following results, the initial detection parameters were as follows.

When using an inductive head, the channel impulse response estimate was set to a dipulse: $\frac{at}{a^2+t^2}$. When using a magneto-resistive head the estimate was an impulse: $\frac{a^2}{a^2+t^2}$. The parameter a was chosen, almost arbitrarily, to approximately match the density of the channel. In a commercial system the value would depend as much on the card reader as on the card itself.

The look-ahead parameter, d', was 0 and the decision delay, d, was 4 bits. The decision feedback had a length of 17 bits (the total buffer length, n, was therefore 22). These parameters were chosen to complement the $(d, k) = (1, 3)$ properties of the code, more than the anticipated inter-symbol interference.

A signal oversampling level of 4 was chosen to aid the velocity tracking mechanism: the total FIR buffer size was therefore 88 samples. The LMS algorithm used a step size of 2×10^{-6}. This value was chosen visibly—large enough

so that the channel estimate would change noticeably by the end of the signal run, but not large enough to cause the channel estimate to look obviously wrong or adapt before the velocity could be locked down. This value was also checked against very long signal runs.

The initial velocity estimate was kept open: several decoding runs at each velocity were tried. The initial phase estimate was not specified even though this introduced some uncertainty into the starting conditions when testing the velocity acquisition.

Although it would appear that there are many arbitrary choices for the testing, a fully automated system could make its own decisions. For example, the initial velocity and phase can be set by a small start sentinel signal before the data. The adaptive filter could be started with a perfect impulse or step function, and then the decoder run backwards through the data so that by the time the pointer is at the start of the signal, the filter has already adapted. Some experimental work was done to choose values for n and d [unpublished] but in the end computation time, code parameters and velocity tracking were of more practical importance than the benefit of larger buffers lowering the bit error rate. On a 166 MHz Pentium system the decoder as laid out above works at about 100 bits/s. This is certainly not fast, but as an unoptimized experimental system it is acceptable (an easy 20-fold increase in speed can be realized by system-specific optimizations).

Figure 5.23 shows simulated (i.e., filtered) waveforms corresponding to different code clock oversampling rates (a measurement, which is inverse to velocity, that will be used frequently). Figure 5.24 shows real waveforms read from a credit card at different bit densities. They were read with the mechanical card reader and an inductive read head.

5.6.2 Real signals, jitter and amplitude

As indicated in section 2.2.3 (page 43), the specialized card-writing head with a wide gap achieves much better amplitude response along a credit card than

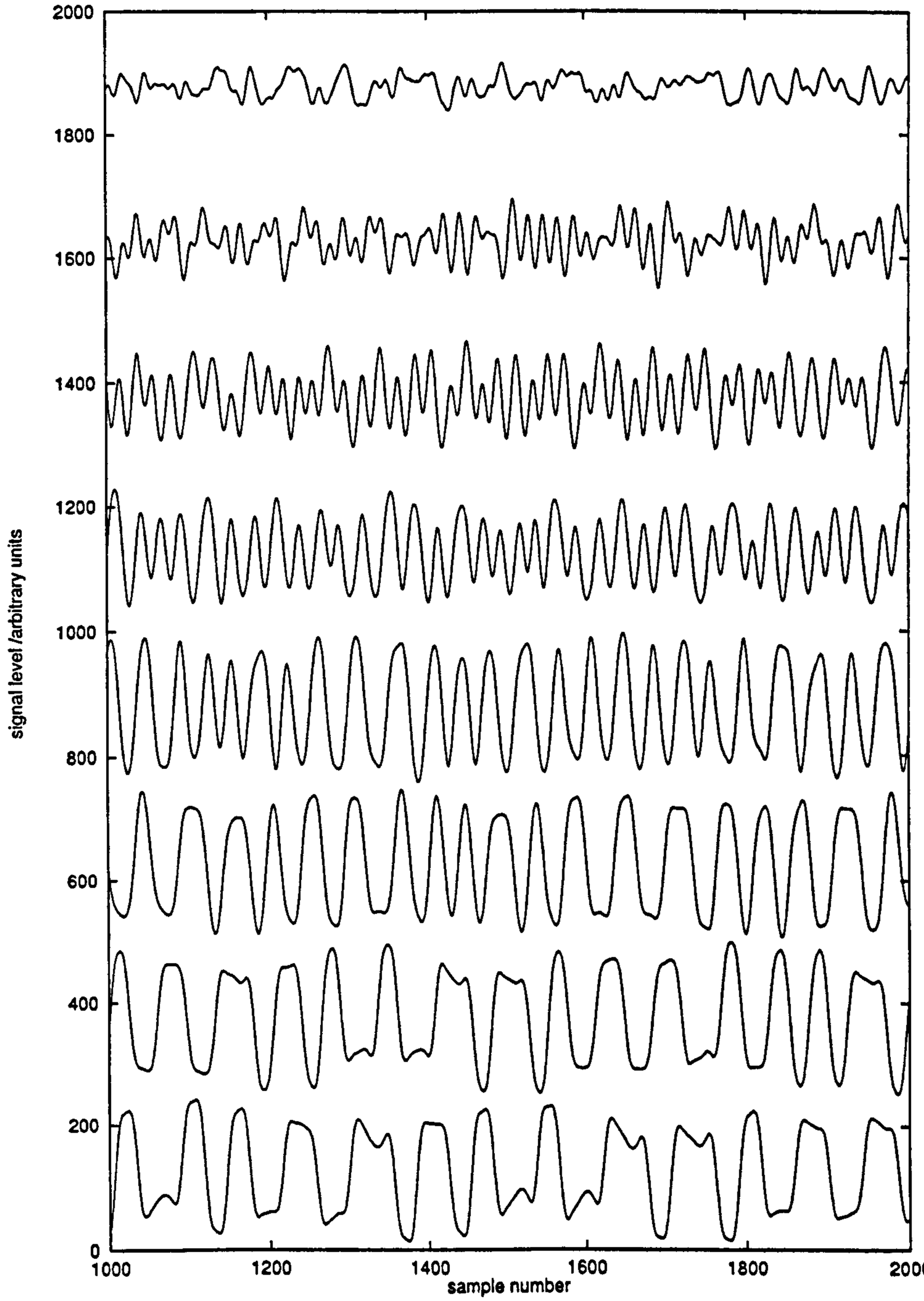


Figure 5.23: Simulated MR head waveforms at different code clock oversampling rates using variable rate code $(d, k) = (1, 3)$. The oversampling factor going from top to bottom: 2.0, 3.0, 4.0, 5.0, 7.0, 9.0, 12.0, 14.0.

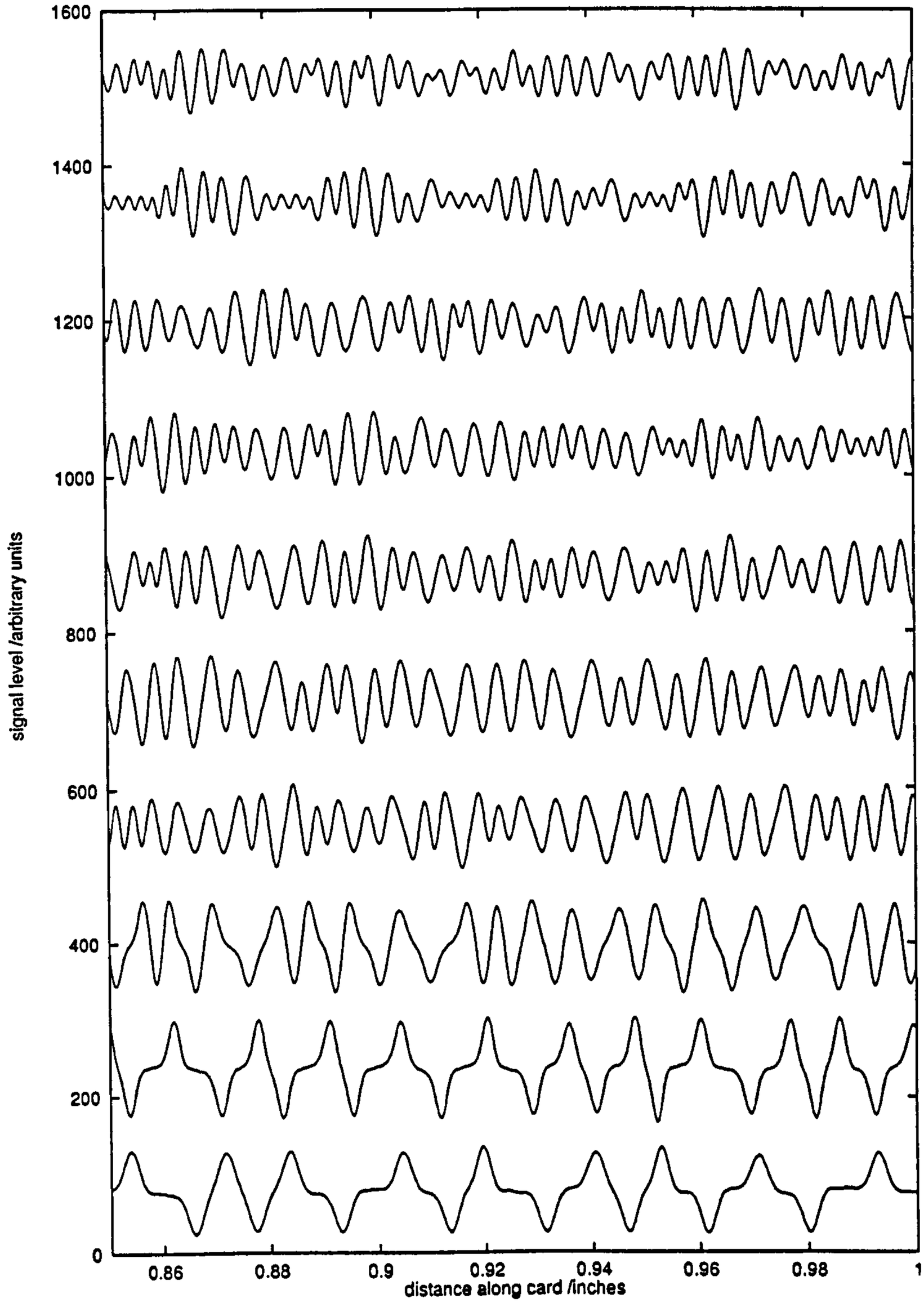


Figure 5.24: Credit card waveforms at different bit densities using variable rate code $(d, k) = (1, 3)$. A mechanical reader was used with an inductive read head. From top to bottom in bits/in: 900, 800, 720, 720, 660, 580, 540, 350, 230, 170.

does a compact cassette audio head. This is because its wider gap allows the magnetic field from the head to penetrate further down from the head into the medium. However, this also has its down side. The credit card is not a thin-film medium: i.e., the medium cannot, on the scale of a bit-length, be considered to be a flat rectangle. Instead it appears as a thin cuboid. The depth of the medium has a distorting effect on the edges of the transitions: they widen, and their longitudinal position becomes less certain. This effect manifests itself as a considerable quantity of apparently random jitter. Enough, in fact, to make recording at high density very difficult with a fully saturated card. This problem was not as apparent with the audio head, because its field's medium penetration was much shallower. The simple solution is to make a credit card with a medium of controlled thickness. However, the results to follow indicate that with a standard, inexpensive and unmodified credit card, full amplitude recording at densities much higher than their designed density is difficult.

5.6.3 Velocity tracking

The first test is how well the variable speed Bayesian detector can acquire and track variable speed signals. The timing phase was neither preset nor adjusted: all phase correction was through the velocity adjustment mechanism.

Figure 5.25 is a set of traces of velocity estimates from simulated signals (refer to figure 5.23) and shows the velocity acquisition and tracking with an unknown initial signal phase. Above an oversampling factor of about 6 the variable speed Bayesian system locks into the velocity very tightly after an initial unsettled period. During the initial time the signal phase is adjusted though the changing velocity estimate until the real and estimated snatches of signal coincide. The chart in figure 5.26 shows that even when the velocity is instantaneously changed, the system will attempt (and would succeed, given enough time) to track these large changes. The ability to track a smaller instantaneous change is less well defined because the $(d, k) = (1, 3)$ code itself allows a certain flexibility. Real velocity changes are continuous, and are not as prone to code

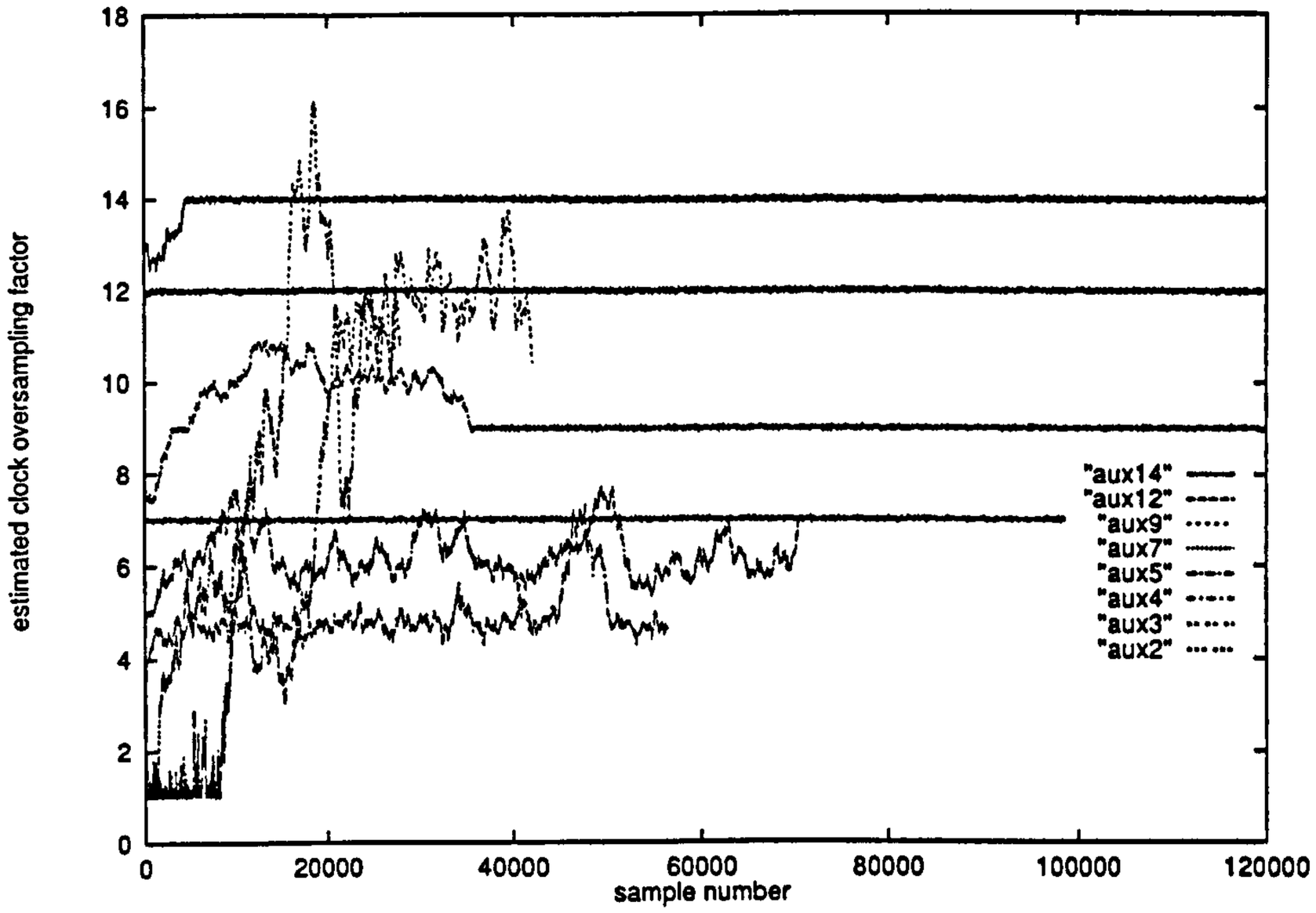


Figure 5.25: Velocity acquisition for simulated signals at oversampling factors of 14.0, 12.0, 9.0, 7.0, 5.0, 4.0, 3.0 and 2.0.

failure mistakes as instant changes.

The charts in figures 5.27, 5.28 and 5.29 show the detector acquiring lock to constant speed real cards with different initial oversampling factor estimates. Figure 5.30 shows that the system can also track the velocity of a hand-swiped card read using an MR head with reasonable accuracy.

Perhaps the most striking or surprising feature is that even when the signal velocity is well beyond the apparent range of the velocity correction, the signal snatch velocity estimation method given in section 4.2.2 (page 84) still makes mostly correct decisions: one might have expected the method to be no better than a 'random walk' with such poor estimates. The second feature is how rapid the convergence is. The system can lock over a wide range of velocities very rapidly: a 50% to 100% speed change can be accommodated over the range of 100 magnetic transitions. Not only that, the convergence appears linear, rather than exponential: i.e., the velocity estimate convergences with the

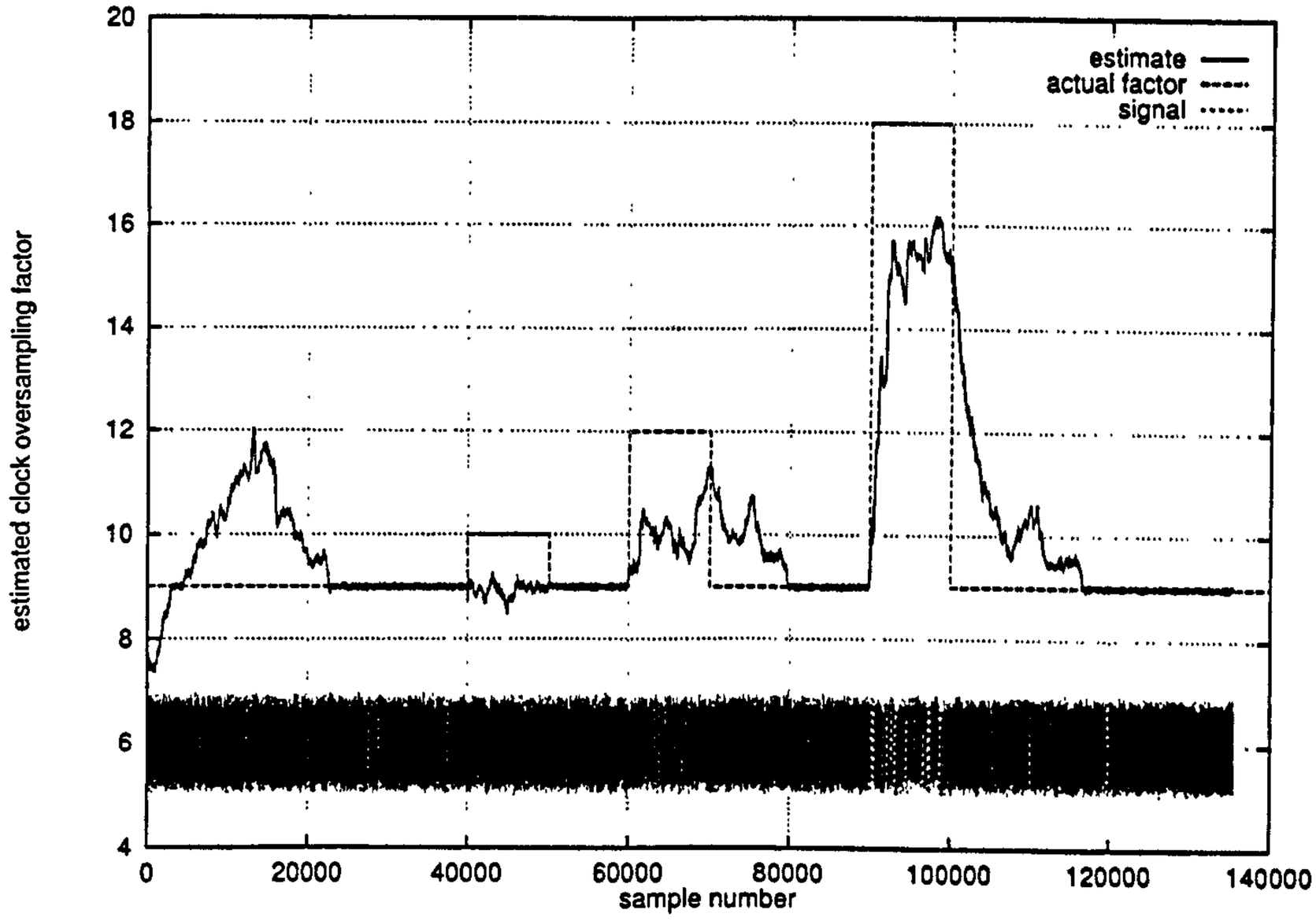


Figure 5.26: Estimated oversampling factor with instantly changing velocities.

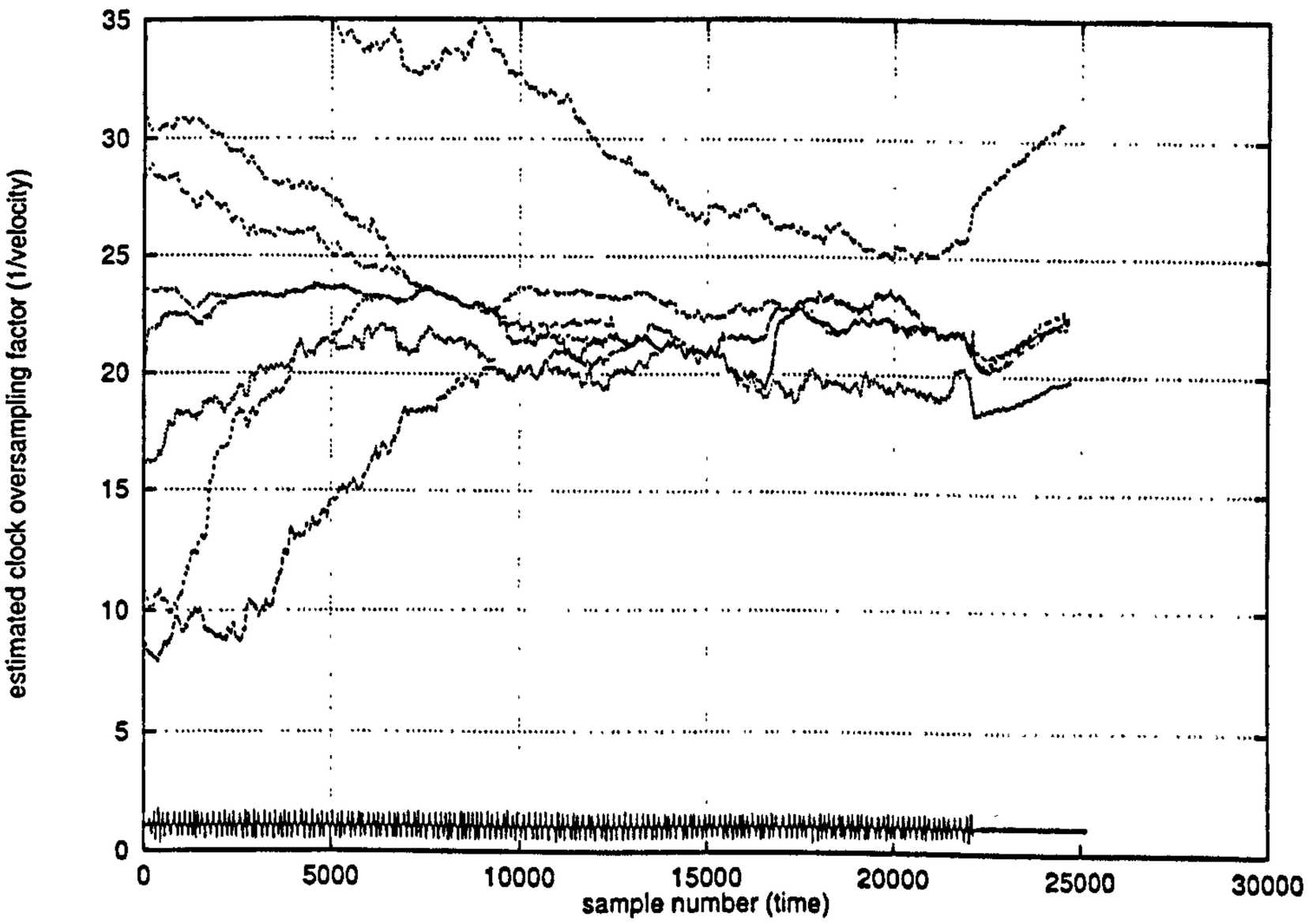


Figure 5.27: Velocity acquisition at 170 bits/in with different initial estimated velocities. The read-back signal is shown at the bottom of the figure.

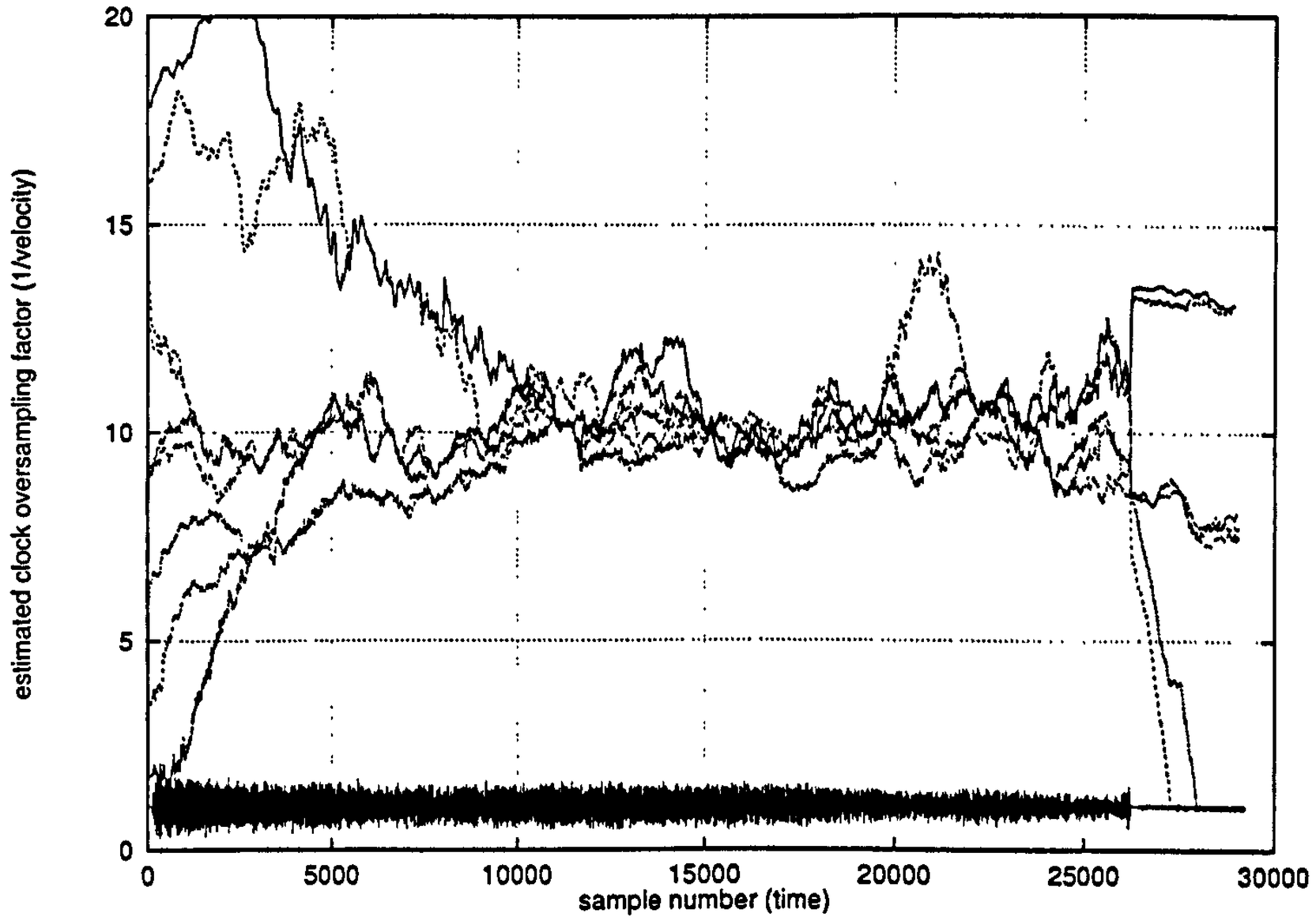


Figure 5.28: Velocity acquisition at 540 bits/in with different initial estimated velocities. The read-back signal is shown at the bottom of the figure.

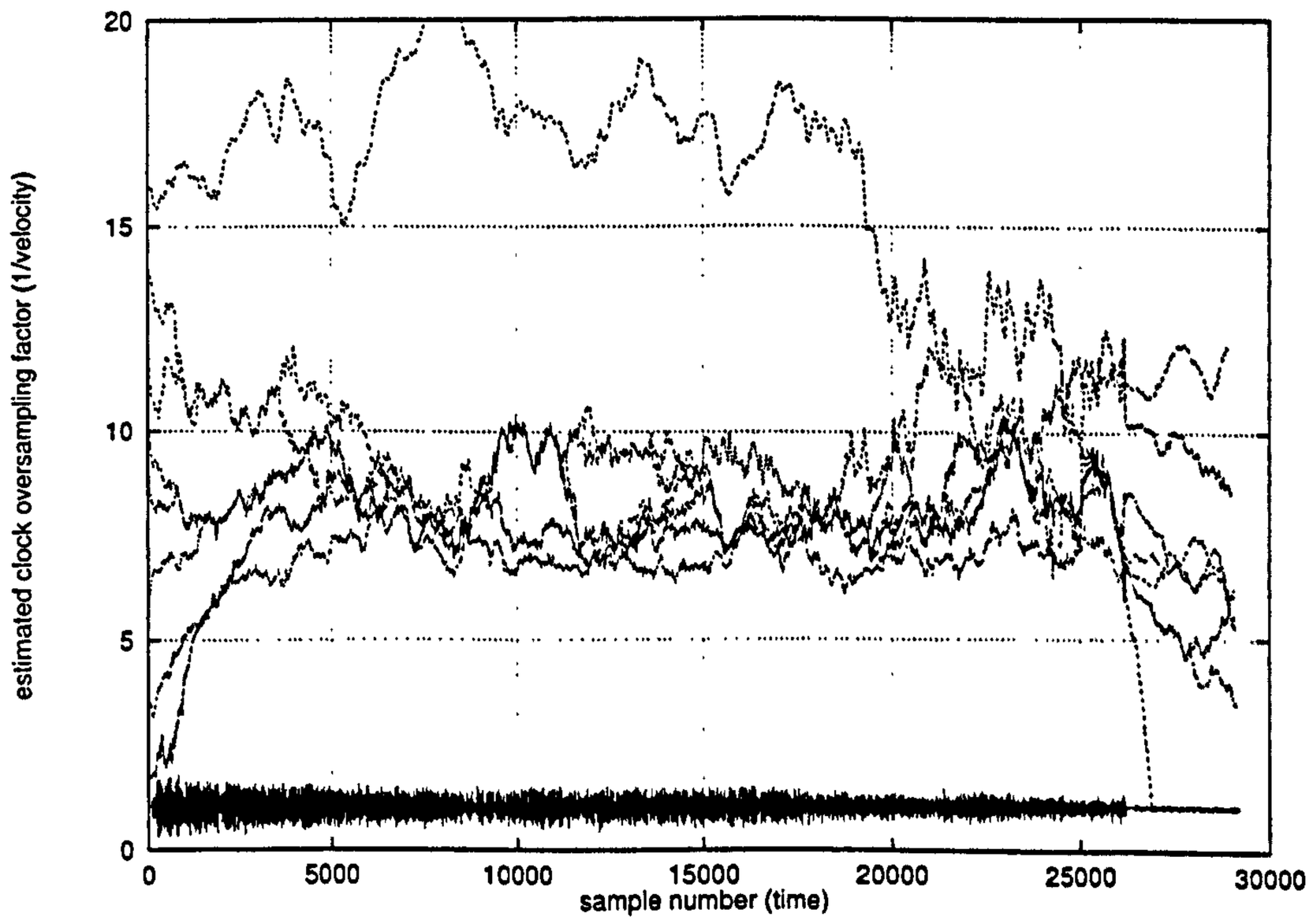


Figure 5.29: Velocity acquisition at 720 bits/in with different initial estimated velocities. The read-back signal is shown at the bottom of the figure.

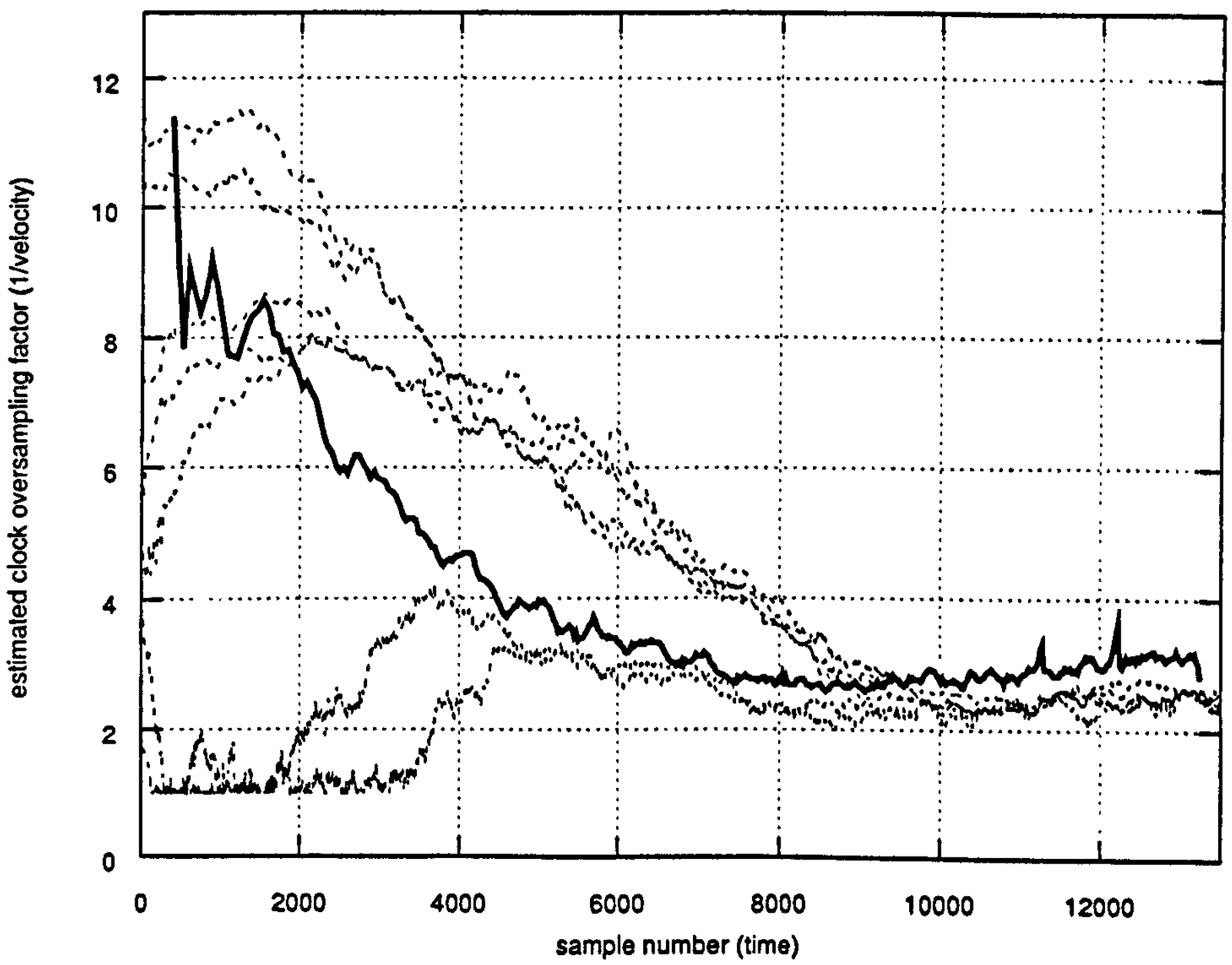


Figure 5.30: Velocity acquisition with a hand-swiped card. The bit density was about 500 bits/in. The thick solid line shows the velocity measured from a separate clock track. An MR read head was used with a Lorentzian initial channel estimate.

true speed and does not stop or slow down until the speeds match. The convergence speed is not unusual or magical (a simple "count the peaks and divide by time" method would find the velocity to the same degree of accuracy in 20–30 magnetic transitions given that the code is reasonably random). However it is undeniably responsive and rapid. It is also interesting to note that the real signals, with their higher jitter content, are slightly easier to acquire than the simulated signals. This is probably down to the noise content: the adaptive feedback filter will adapt to match a perfect signal, even when that signal is in fact wrong—the extra noise and uncertainty in the real signals appears to be an advantage: there being enough 'thermal' noise to allow the rejection of meta-stable local 'energy' minima.

Unfortunately it is also undeniable that the variable speed Bayesian detector makes velocity mistakes. One would not expect the velocity traces to be perfectly aligned: the adaptive feedback filter and phase offsets are often non-degenerate on runs with differing starting conditions. While the velocities do fluctuate, the margin of difference is between 10 and 30%. A clocking error corresponds to a 25% error range in a $(d, k) = (1, 3)$ code. This implies that the detector is making bit errors where the velocity error exceeds this level, and that the detector may be making half-bit errors when the difference is 12%. However, the estimate alternating between too fast and too slow would remove some of these cumulative errors (such a resonance could be caused by a particular phase alignment) but no regular alternating pattern is evident here. This suggests that jitter is confusing the velocity update mechanism, as transition timing jitter will manifest itself as a velocity jitter. In figure 5.28 this may have caused the timing estimate for one run to settle at the wrong clock rate (shown in closer detail in figure 5.31); but later correct itself. In this situation the rapid velocity adjustment is working against the system's stability. Another error source that can affect velocity estimation is amplitude variation. This is a weak secondary source and has not been studied in detail.

The choice of initial velocity has a non-deterministic and fairly long-term effect on the operation of the detector. Some very poor initial velocity estimates

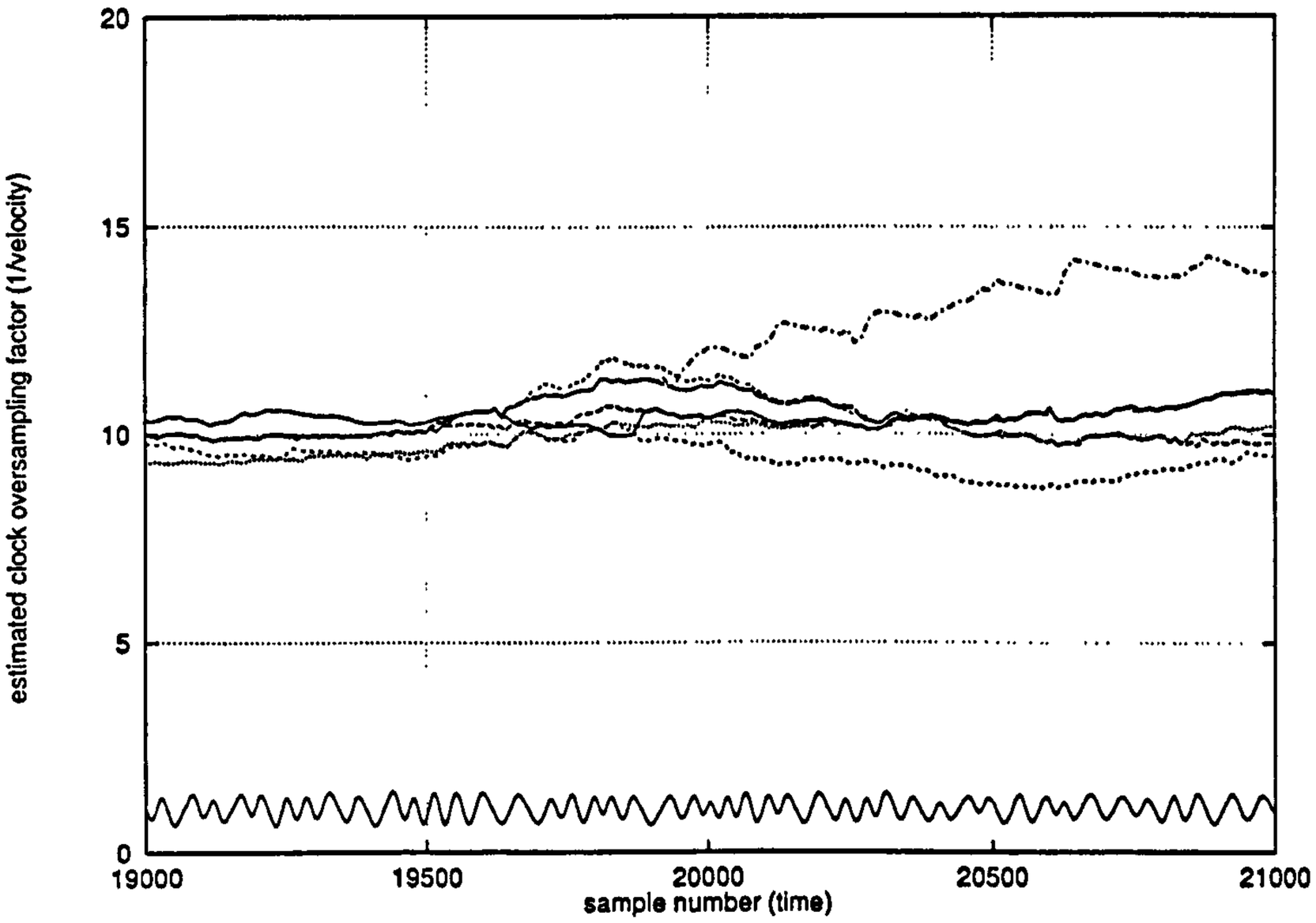


Figure 5.31: Velocity acquisition close-up at 540 bits/in with different initial estimated velocities. The read-back signal is shown at the bottom of the figure.

can damage the adaptive filter setting so badly that recovery is not possible. Examples are evident in figures 5.27 and 5.29. This is expected, although it is not known whether the ailing velocity estimator will recover in the long term. The manifestation of the oversampling estimate that is too high is that the adaptive filter becomes very detailed. The Bayesian detector will then maximize its probability by fitting spikes in the transverse filter coefficients to apparent spikes in the signal. The converse can also happen with an oversampling estimate that is too low. In this situation the adaptive filter becomes broad, and the Bayesian detector reconciles this by slowing down the oversampling to simply match the duty ratio of the filtered code to the level of the stretched signal. This is a metastable state that is often broken; it is also less likely with an initially 'DC-free' channel estimate in the filter.

A final note is that the system appears to work better at mid-densities; rather than very high or very low densities. While poor performance at the high den-

sities is easy to understand (the bit cells are very small and almost buried under noise) the low density performance is perhaps puzzling at first. However there are fewer transitions to lock into at low densities, hence the adaptive filters will appear to be poorer and noise is stronger relative to (the scarce) signal.

5.6.4 Confidence margin

The Bayesian detector relies on a probability calculation to make decisions. This probability is directly related to the confidence that the detector places on that decision. While this technique has lead to useful results in error margin analysis [54] it is of less interest in a high-error environment. Figure 5.32 shows the non-normalized probability value for two simulated signals and figures 5.33 and 5.34 show traces from real cards at constant velocity: their forms may not be of immediate significance but there are several points to note.

Firstly the signal appears very noisy. This is partially because these traces show $\sum_i^{\text{all valid } i} \eta_k^{(i)}$, rather than $\sum_i^{\text{all } i} \eta_k^{(i)} \eta_k'^{(i)}$ as in equation (5.31). Indeed this is why the simulated signal confidence is an apparently noisy value. In the real traces the detector is also making errors and looking at jitter; both hindering the adaptive filter and dragging the confidence down to zero from time to time.

The second point is that the confidence (or peak confidence) increases slowly over the length of the trace. This is expected because the channel estimate is adapting to the channel, and so the signal match improves.

Finally, the confidence drops, and then rises rapidly when the signal has stopped. This is also not surprising: the confidence drops because the filtered estimate signal can no longer be fitted to the zero-valued read-back trace, but it rises quickly as the adaptive filter widens and decreases in amplitude—it would eventually perfectly match a pure zero signal.

5.6.5 Bit error rate

In traditional systems the bit error rate is easy to measure: write a bit pattern, read it back and bits that don't match are in error. When a significant amount of

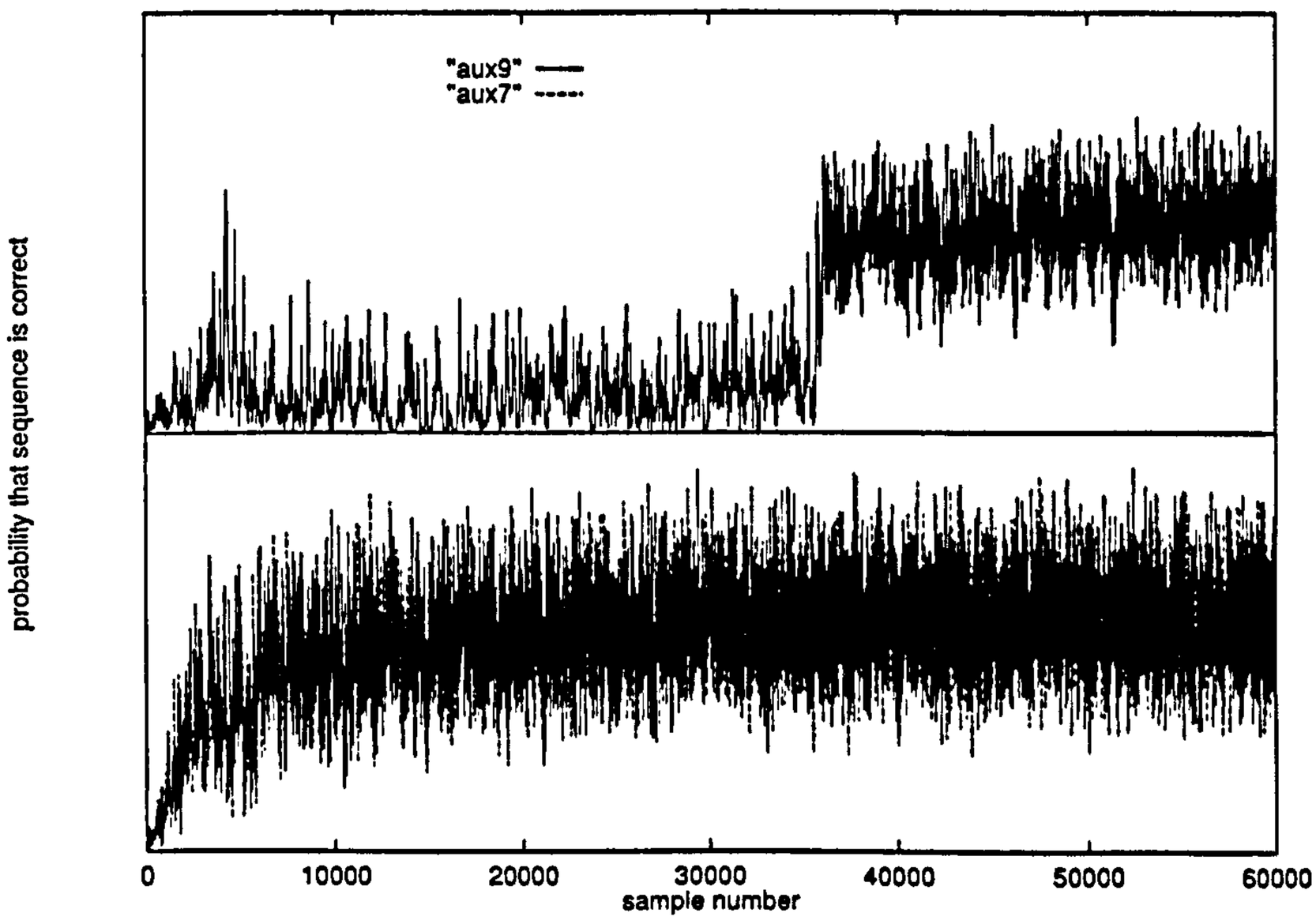


Figure 5.32: Confidence margin (non-normalized probability of estimated and real signals matching) of two of the sample traces shown in figure 5.25 at oversampling factors of 9.0 (upper) and 7.0 (lower). This trace is η_k with no correction for the a priori probabilities, η'_k.

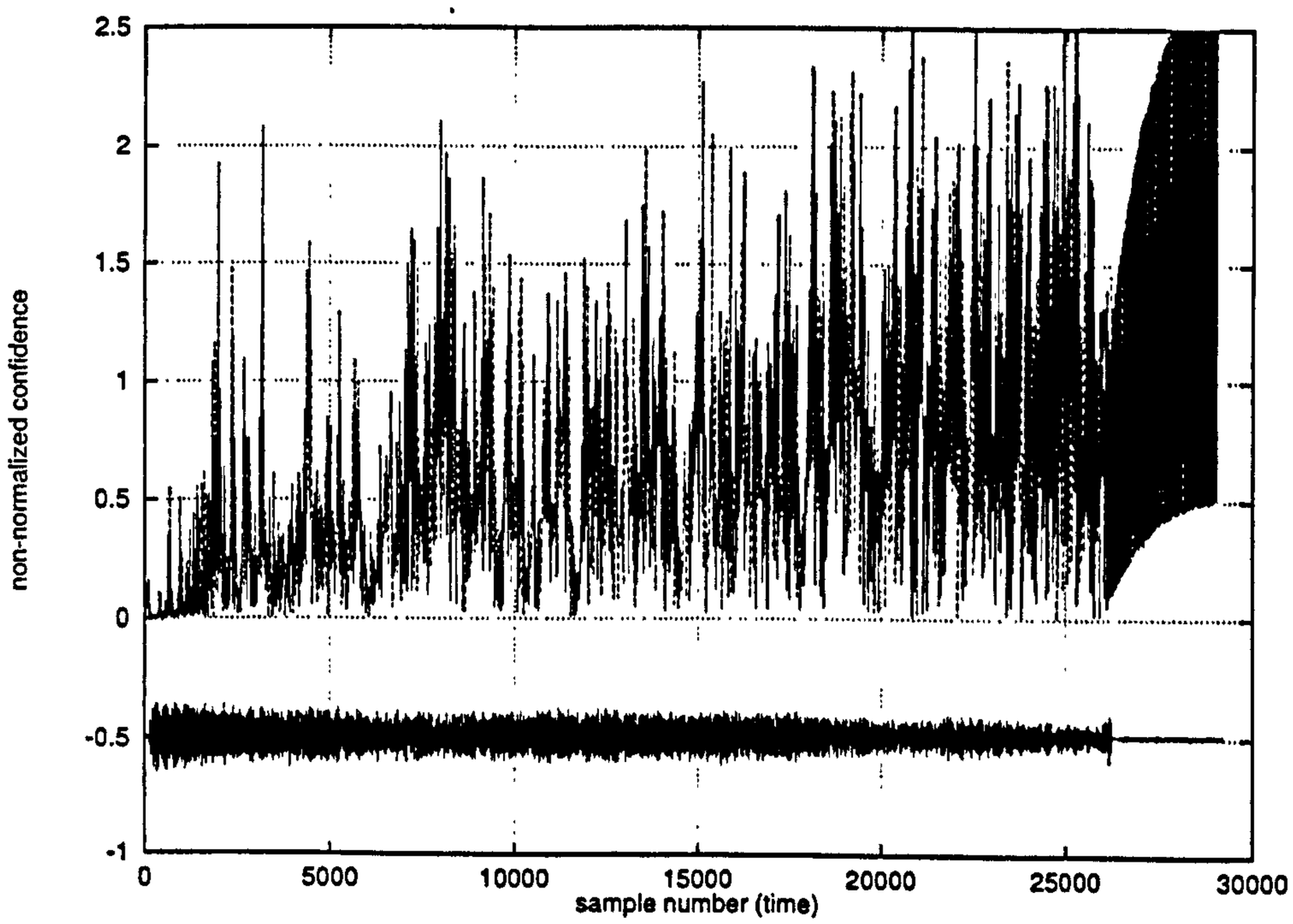


Figure 5.33: Confidence margin (non-normalized probability of being correct) of the signal trace shown in figure 5.28 starting at initial oversampling estimate 2.0. This trace is η_k with no correction for the a priori probabilities, η'_k. The read-back signal is shown beneath.

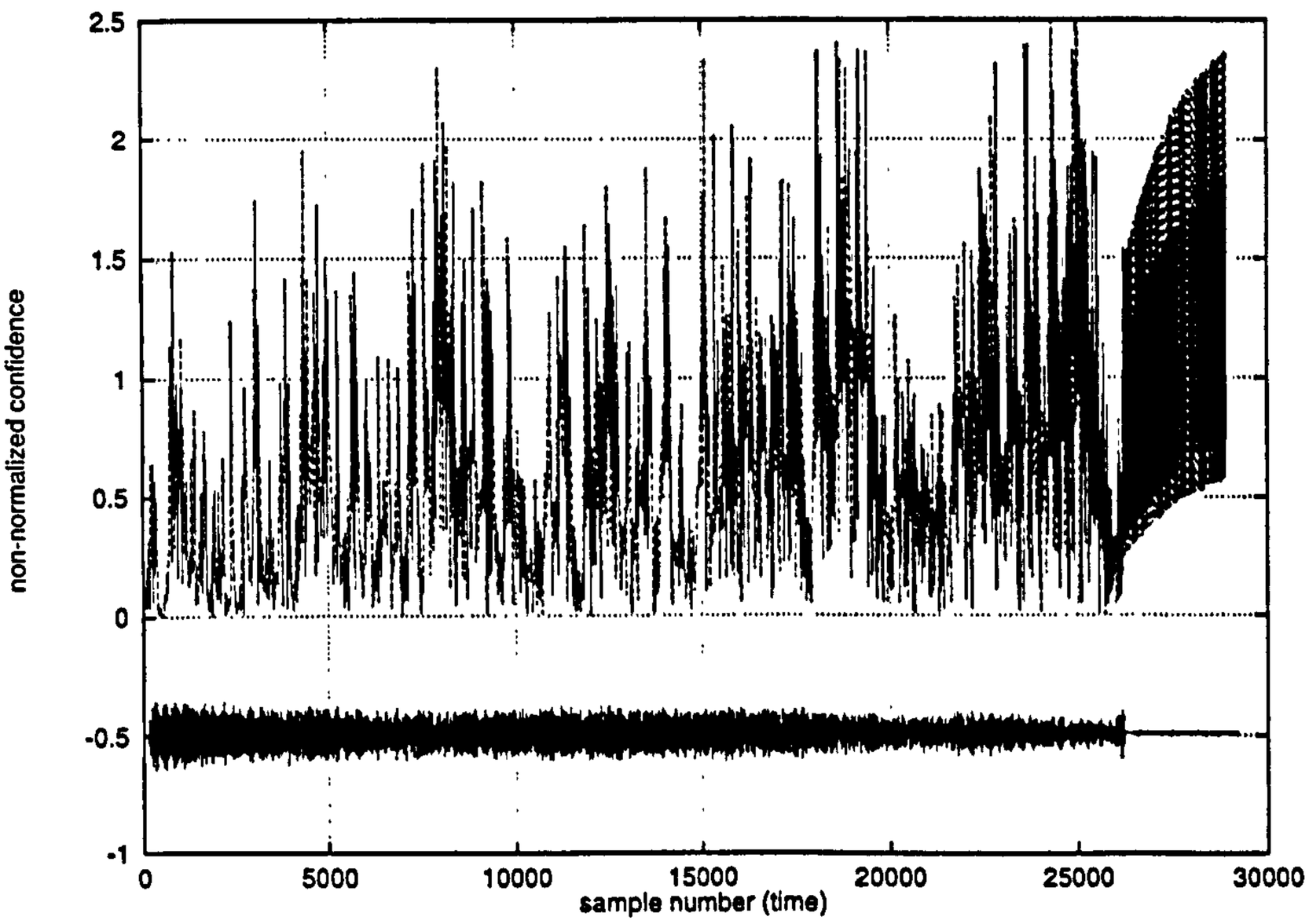


Figure 5.34: Confidence margin of the signal trace shown in figure 5.28 starting at initial oversampling estimate 10.0, showing a reduced confidence level in the velocity estimate divergence region. The read-back signal is shown beneath.

jitter and velocity uncertainty is factored in however, the measurement becomes more philosophical. If two correctly decoded stretches of bits are separated by nonsense, should the total correct bits count or merely the longest run? Should a missing bit count as one error, two errors, or many errors? While contemporary decoding routines treat a *single* missing bit as many errors, one cannot discount the possibility that future decoding systems will be tolerant towards occasional missing bits.

The above concerns are direct consequences of using a variable speed system with a short, fixed length of recording medium. Using straightforward frequency modulation (FM) code and variable rate code in a constant velocity card reader the bit error rate can be measured in standard terms if a repeating pattern is used: this way missing regions are not so critical as when the output is decoded correctly again the data pattern position can be quickly recalculated. The results of using a peak detector on this simplified channel are shown in figure 5.35. (The high error rate at low density was because there were fewer bits stored on the card, and the data pattern was not framed; therefore the first and last few bits were nearly always in error.) The signals for the figure were also written with the audio cassette head, which does not fully penetrate the card's recording medium and tends to reduce jitter at the expense of a weaker read-back signal. The velocity following was based on simple inter-peak timing (section 4.1.2, page 77) because continually repeating short patterns are not amenable to manipulation by an adaptive system: an adaptive system would adapt to the pattern on the channel in preference to the channel alone. Figure 5.35 may be considered to be reasonably close to the practical bit density recording limits available to a credit card reader.

For the more realistic pseudo-random bit patterns, the inter-peak timing method fails because of the (now) unpredictable effects of inter-symbol interference preventing consistent synchronization. To present the results with the variable speed Bayesian detector, the following procedure was used (see figure 5.36). A card was recorded and read back. The signal was passed through the Bayesian detector and decoded. This decoded data stream was compared

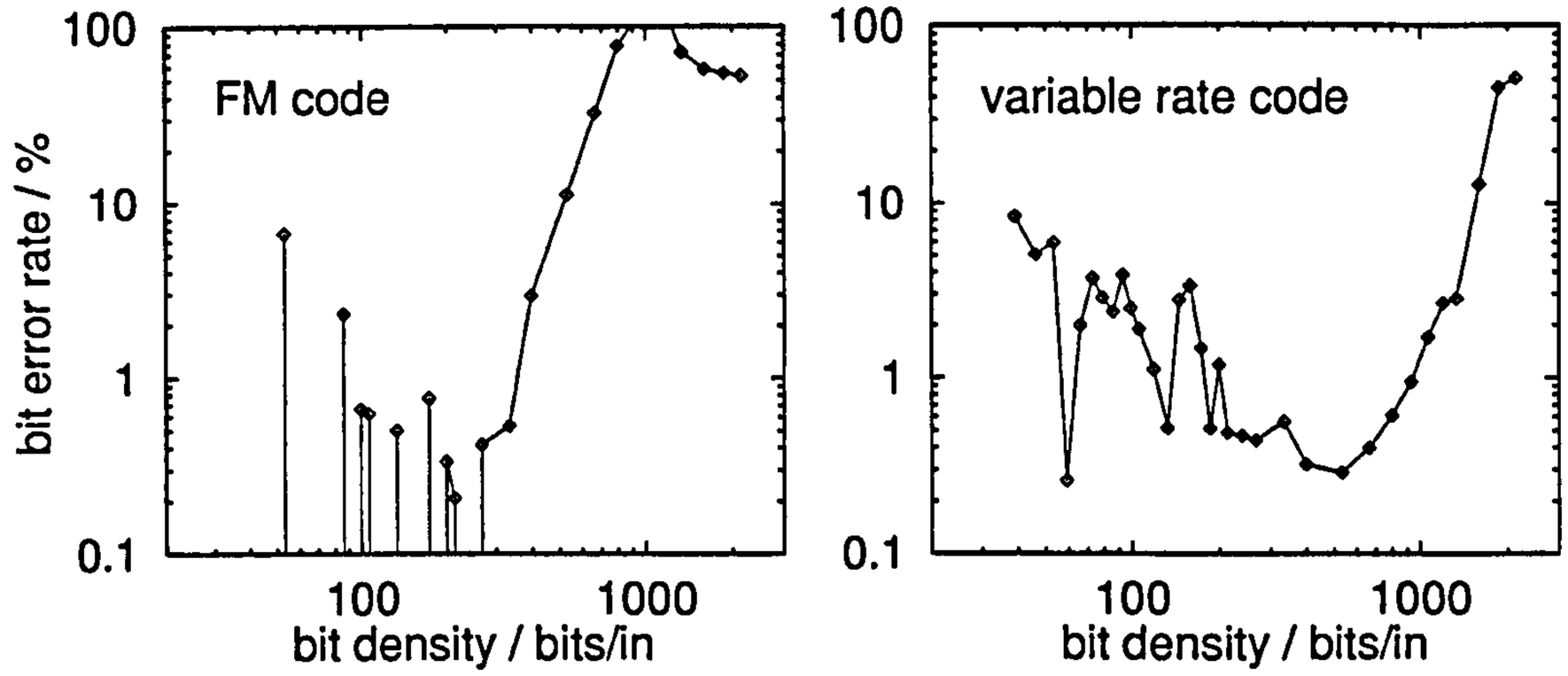


Figure 5.35: Error rates for peak detection using FM and variable rate codes. Read with an inductive head reader at constant velocities with simple inter-peak timing updates. A repeating data sequence was used.

with the original by a correlation operation. (In this test, the start of data was not marked on the card, so the signal starts at an undetermined point in the pseudo-random sequence.) The ratio of the peak value from the correlation to the total number of decoded data symbols gives a measure of the correctly decoded bit ratio.

Because the variable speed Bayesian decoder does not immediately lock into the signal parameters (the inter-peak timing method only required 6 transitions in the repeated pattern) and because of the nature of the correlation function, the results in figures 5.37 and 5.38 are not quite straightforward to interpret.

First the correlation function itself is pessimistic, although as a system-wide error estimator it is probably realistic. The correlation functions from decoded tracks do not have a single peak. Consider, for example, the correlation result at 540 bits/in which is shown in figure 5.39. This has a broad positive swathe, implying that the decoder is producing many small correct bit streams but that there have been erroneously missing or additional bits between those correct parts.

Secondly, the correlation has no regard for settling time. If the decoder takes

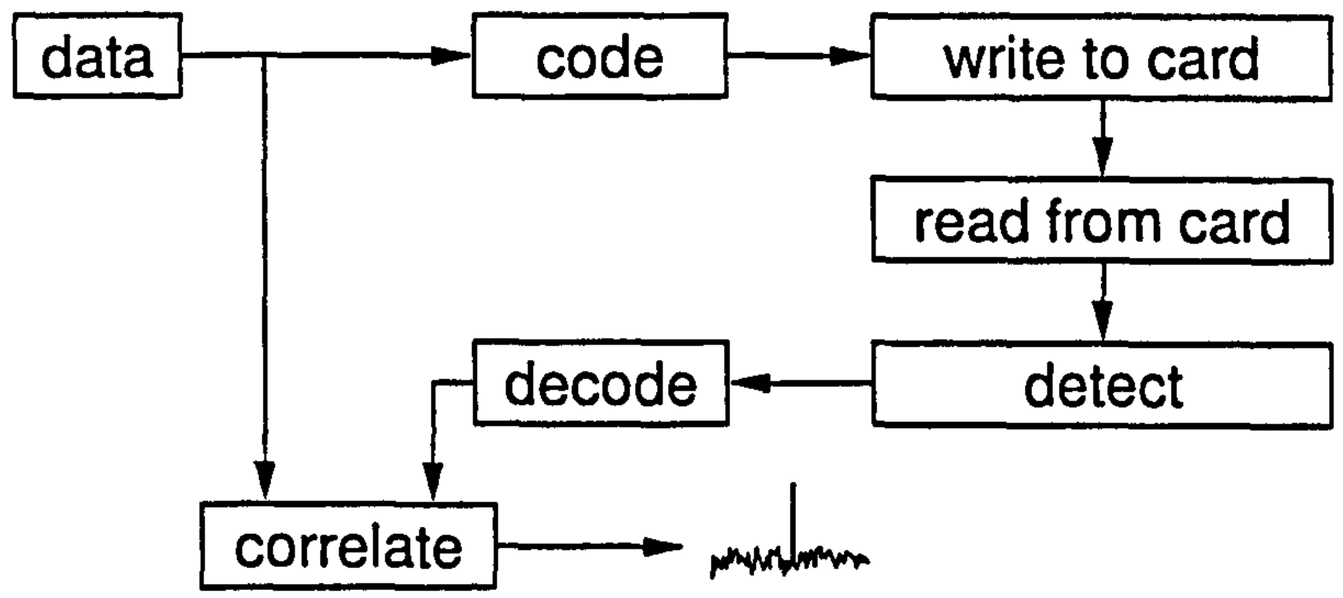


Figure 5.36: Procedure for correlative calculation of bit error rate.

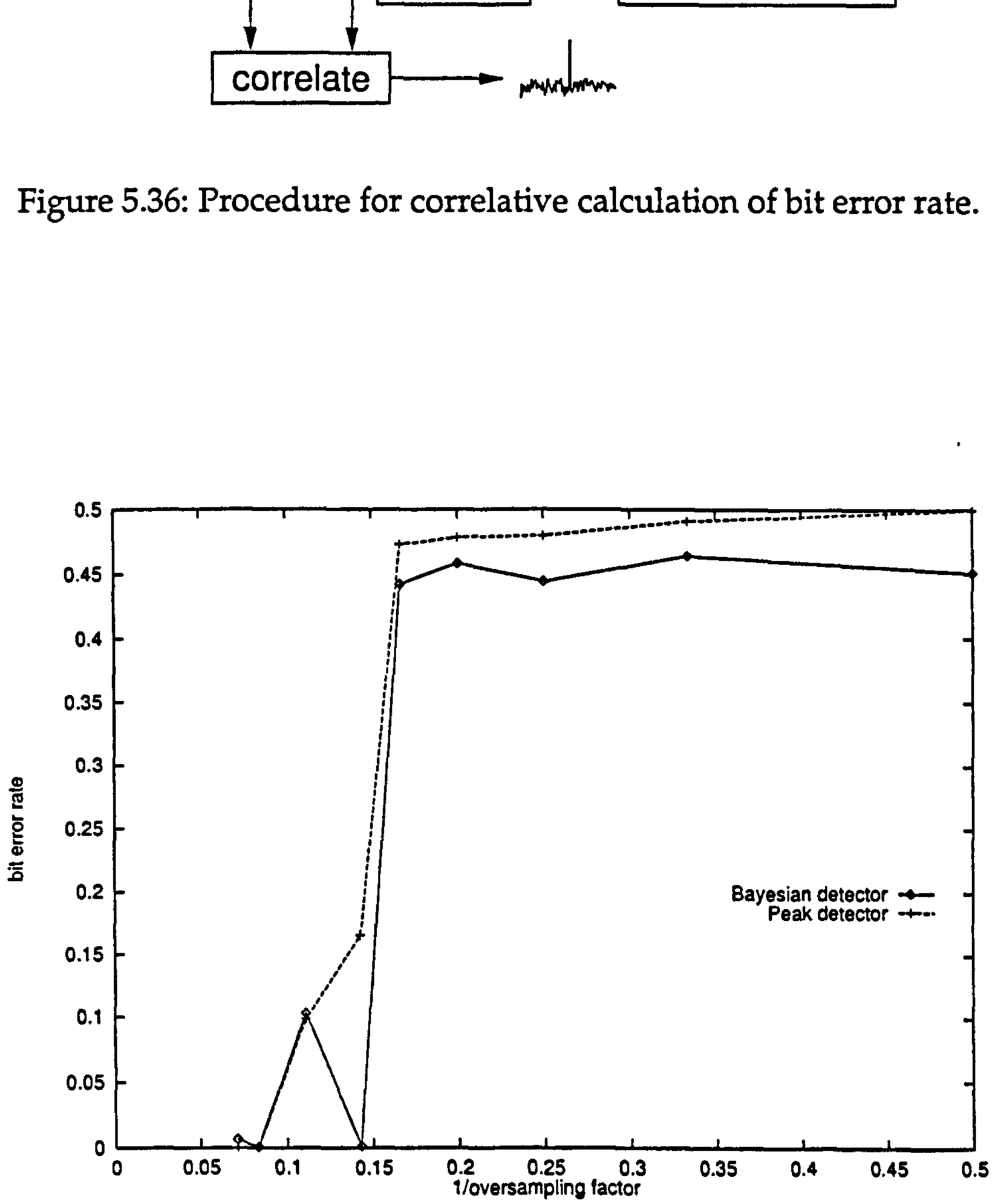


Figure 5.37: Simulated bit error rates for pseudo-random sequences through a Bayesian and peak detector.

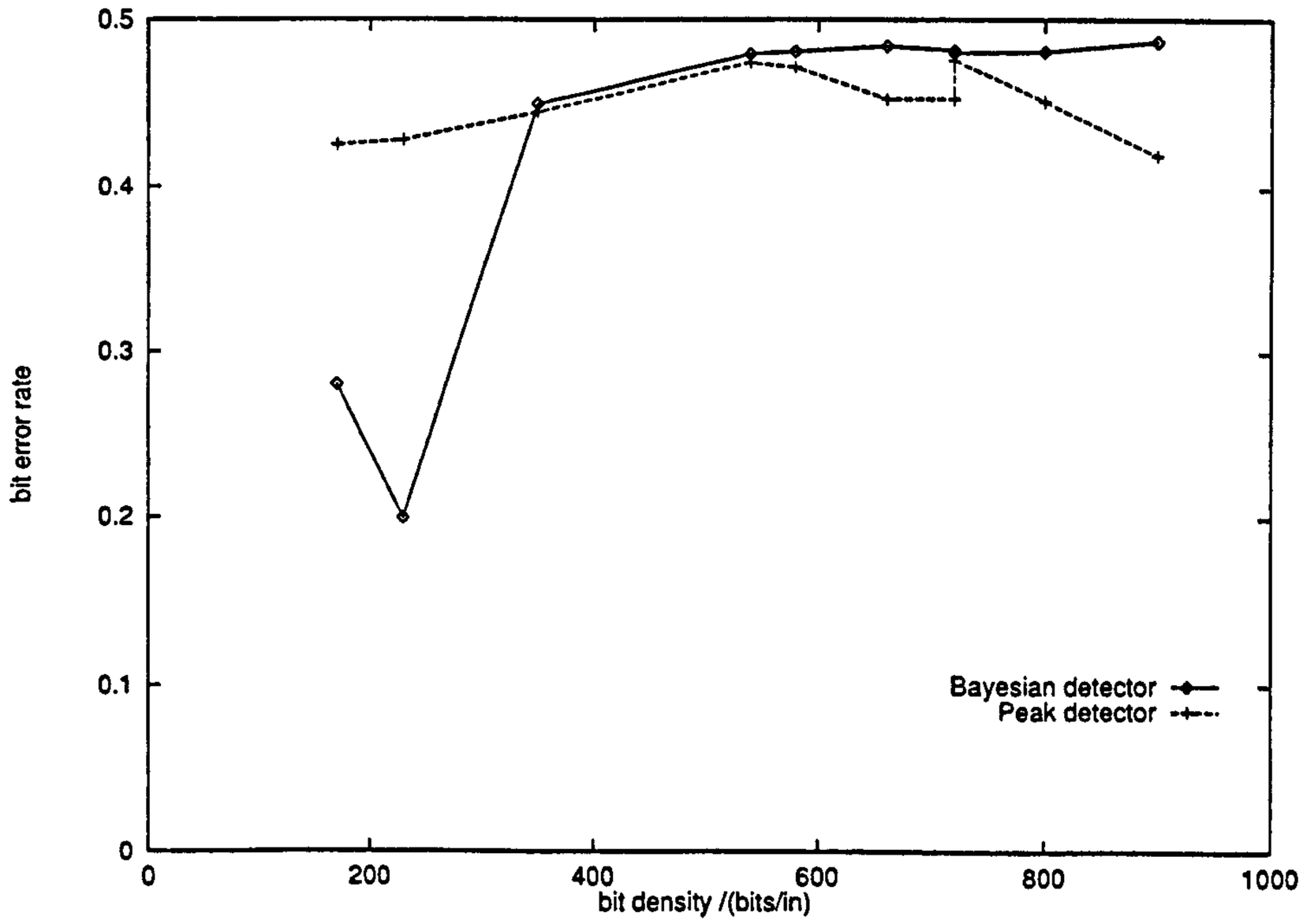


Figure 5.38: Bit error rates for pseudo-random sequences recorded on cards through a Bayesian and peak detector.

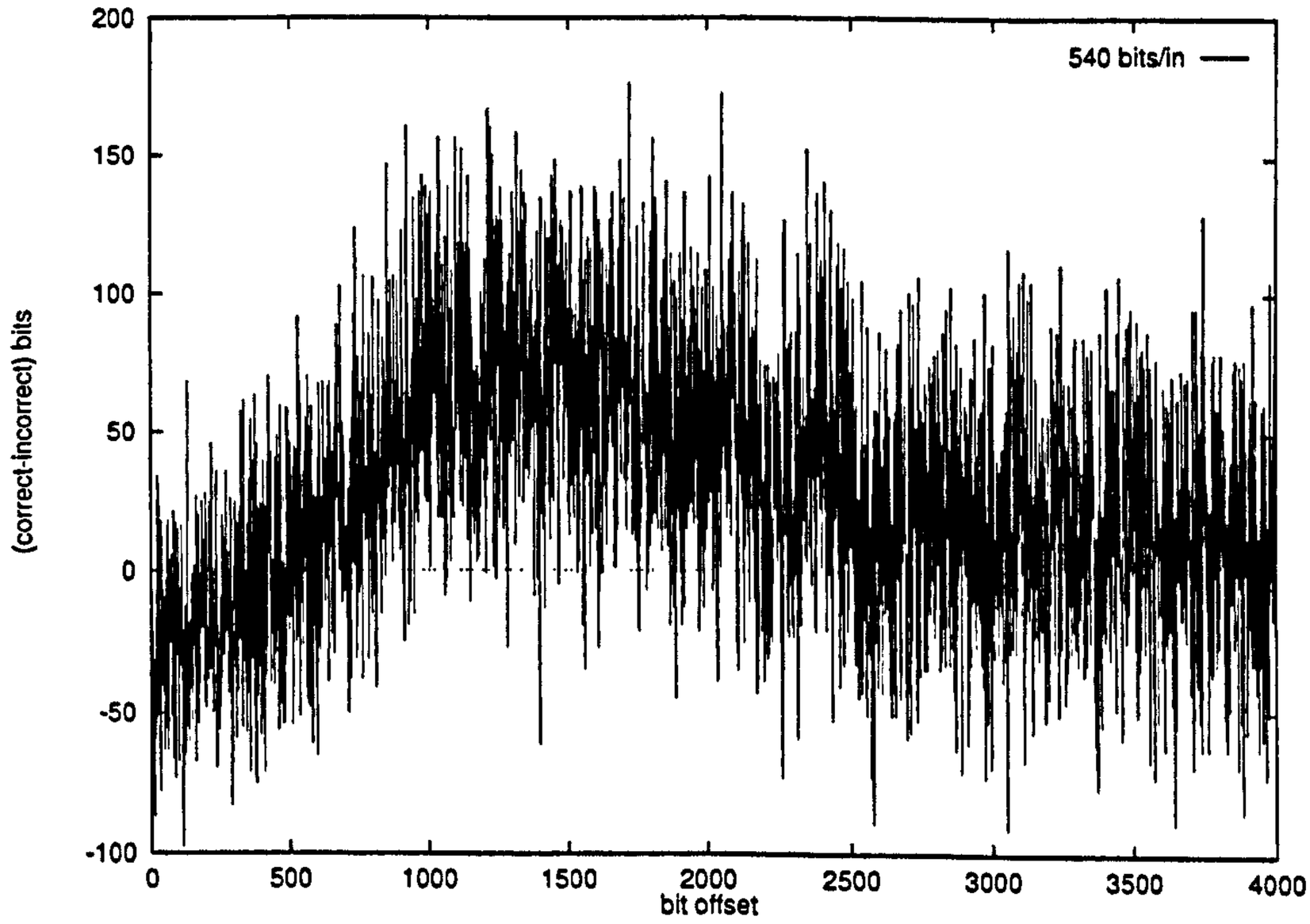


Figure 5.39: Correlation function from decoded track at 540 bits/in.

half the length of the card to settle, and then decodes perfectly, a 25% bit error rate will be recorded (comprising 50% for the first half and 0% for the second half). This is especially prevalent at low densities.

Thirdly, the cards measured for figure 5.38 were written with the wide-gapped card writing head. This increases the amount of jitter in the signal very substantially. This will be discussed further in the next section.

Finally, the cut-off from low error rate to high error rate is very sharp. This is not unexpected in the Bayesian system. Once the bit error rate increases the adaption feature starts to work against the detector. With the velocity tracking in place and also contributing to mistakes, breakdown of the system is more likely. However, the peak detector also has a fairly sharp curve in the simulated (pure linear superposition) densities. The peak detector fails badly with the real pseudo-random data: this is unexpected as the differences between the repeated patterns and pseudo-random patterns are not that great when compared visually. Once again, the disparity between figures 5.35 and 5.38 must be attributed to the jitter problem.

There appears to be little to choose between the variable speed Bayesian detector and the peak detector in terms of bit error rate on a card. The Bayesian detector does have a significant edge, but at densities so low as to be unexciting. It might be generously given a twofold density advantage.

5.6.6 The causes of jitter

Jitter is the uncertainty in the timing of a signal at the scale of a transition length. Jitter can be caused by vibration (perhaps an audible 'screech' as the card goes under the head or by the natural step length when the card is written using a stepper motor). Jitter can also be caused by variations in particle size and coercivity. For example, the magnetic field from a fully magnetized large particle will swamp the field from a nearby smaller particle affecting both the write process (although this effect should be minimal if the field is strong enough) and the read process.

Why should this effect be prevalent in a card written with a wide gap, but much smaller in a card written with a gap too narrow to fully penetrate the medium? There are two immediately apparent reasons. Consider, for example, a medium with many small, evenly-sized particles near the medium surface and rarer large (high coercivity) particles deeper in the medium. A weak write field will magnetize only the small surface particles, leaving the deeper particles unaffected. Not only does this reduce the size of the PW_{50} (a smaller head-medium separation) but also the smaller particles return a more consistent and highly defined field edge.

The second reason is to do with the physical size of the transition when the card is being written. The larger gap will have a larger transition length along and down through the medium.

The jitter seen in the signals from the cards comes from a non-systematic source. This rules out the well-known phenomenon of non-linear transition shift. This was determined by classifying transition lengths according to the transition length prior (figure 5.40) and following (figure 5.41) the transition in question. The figures show no evidence of any tightening of the interval bands. A zero-jitter system would have three sharp bands 2, 3 and 4 times ~67 arbitrary units. The individual transition peaks were spaced sufficiently that inter-symbol interference was avoided.

The sources of jitter above are speculative, and it is not known precisely why the problem is so overwhelmingly large in these particular results.

5.6.7 Variation of parameters

In the variable speed Bayesian decoder there are a number of tunable parameters: most have already been discussed in some form or another. The visible parameters are set out below and the referenced charts show some of the effects of these variations on estimated velocity with a real credit card signal obtained with the magneto-resistive head. Also shown in figure 5.42 is a trace of the velocity as measured by smoothing the inter-peak time from a zero-crossing

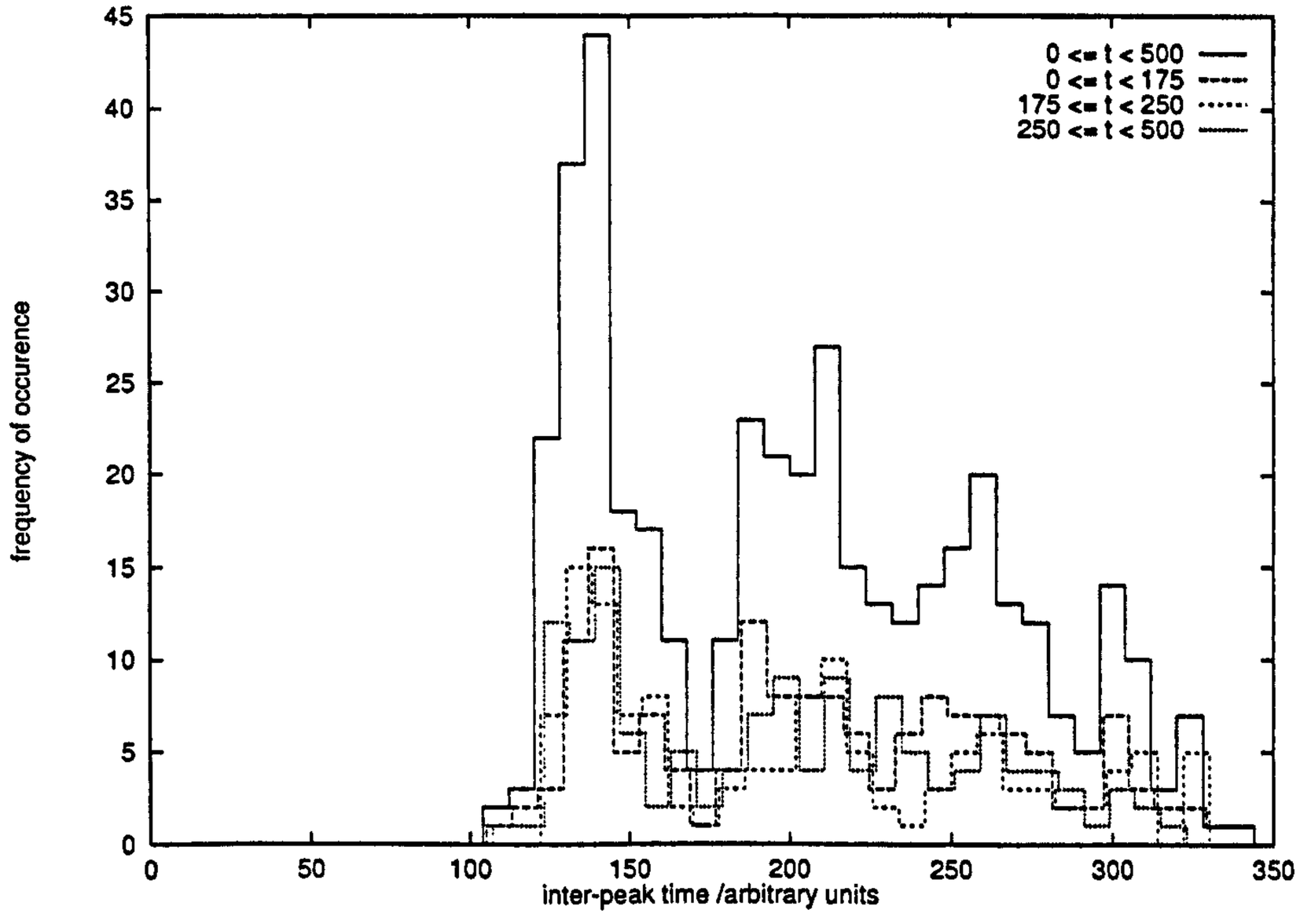


Figure 5.40: Inter-peak timings classified by the prior inter-peak time (t).

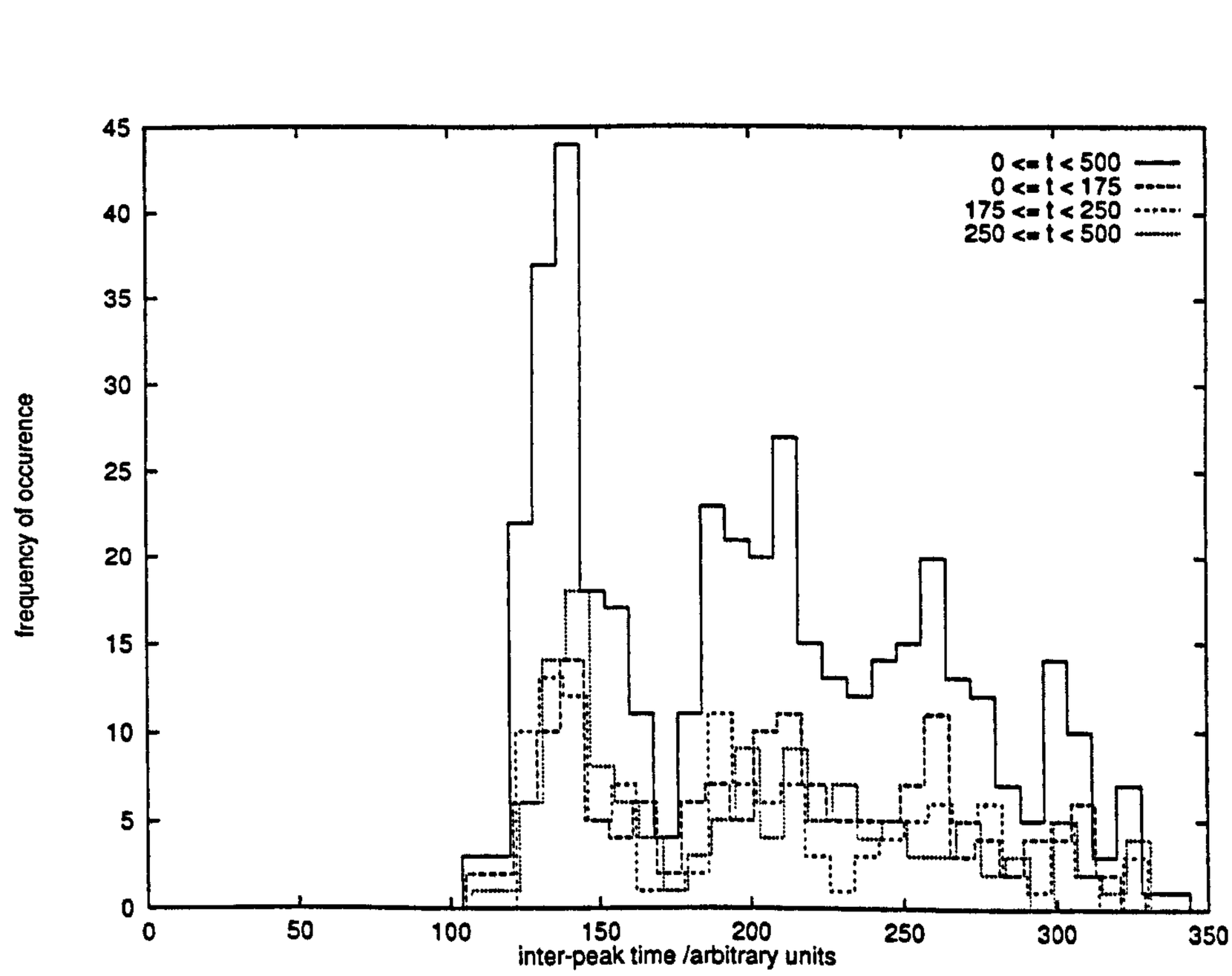


Figure 5.41: Inter-peak timings classified by the following inter-peak time (t).

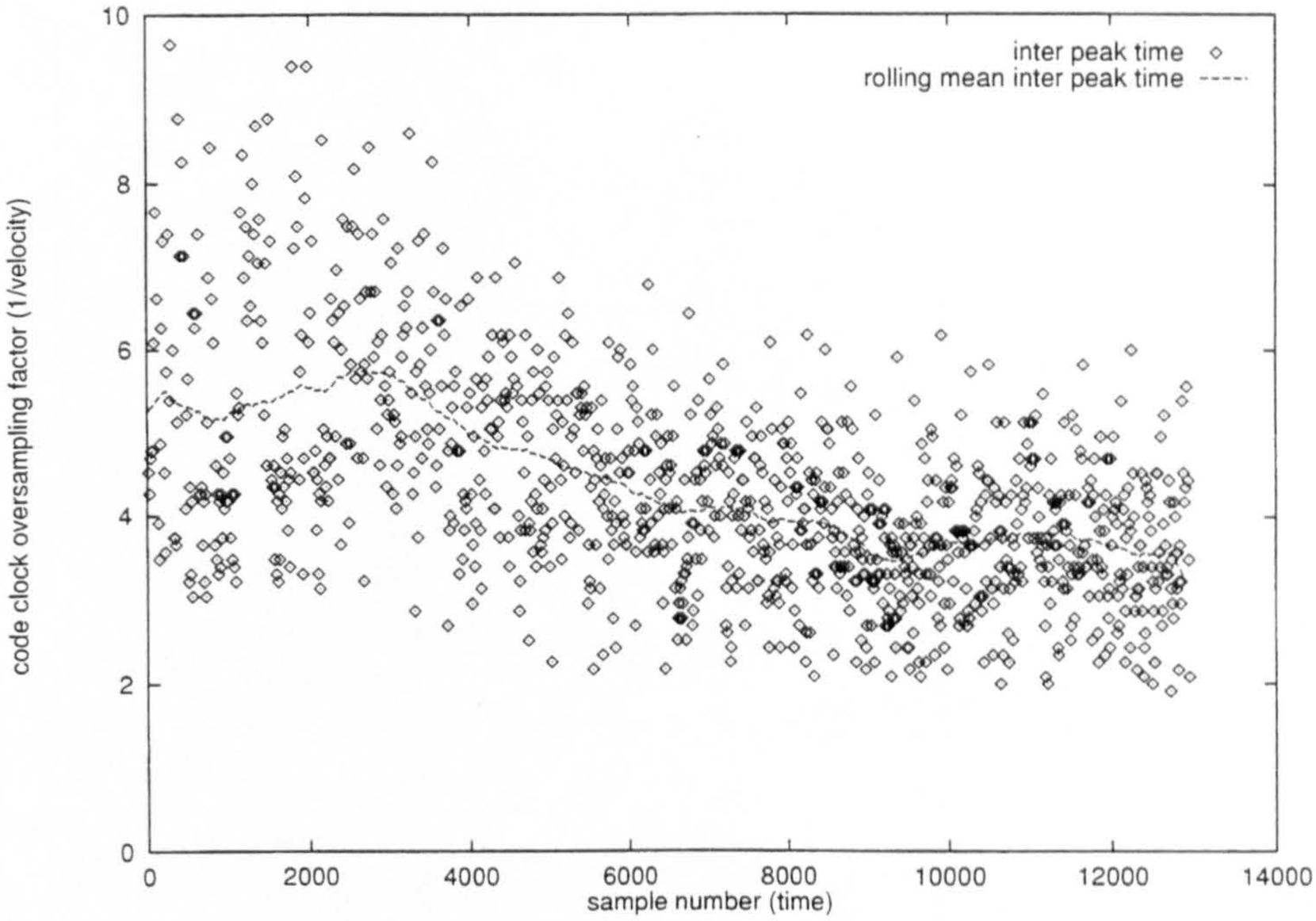


Figure 5.42: Peak detector and mean inter-peak time for swiped card.

detector applied to the signal (a portion of which is shown in figure 5.43).

- Non-linear truncation. The non-linearity correction for the MR head is a free choice of function. Truncation was selected, with two tunable parameters: the upper and lower threshold. These are selected on the basis of measurable signal distortion. Figure 5.44 shows the effects of symmetric strong, appropriate and no truncation. Truncation that is too weak or too aggressive has detrimental effects on the tracking ability, as would be expected. In fact, in the latter case this appears to be due to errors growing in the channel estimate, while the former case is attributable to the signal distortion itself.

- Phase versus velocity correction. Either phase or velocity correction can be calculated with the snatch estimation method. It is also possible to update both phase and velocity simultaneously: by ascribing a proportion of the correction (equations (4.14) and (4.20) in chapter 4) to correct phase

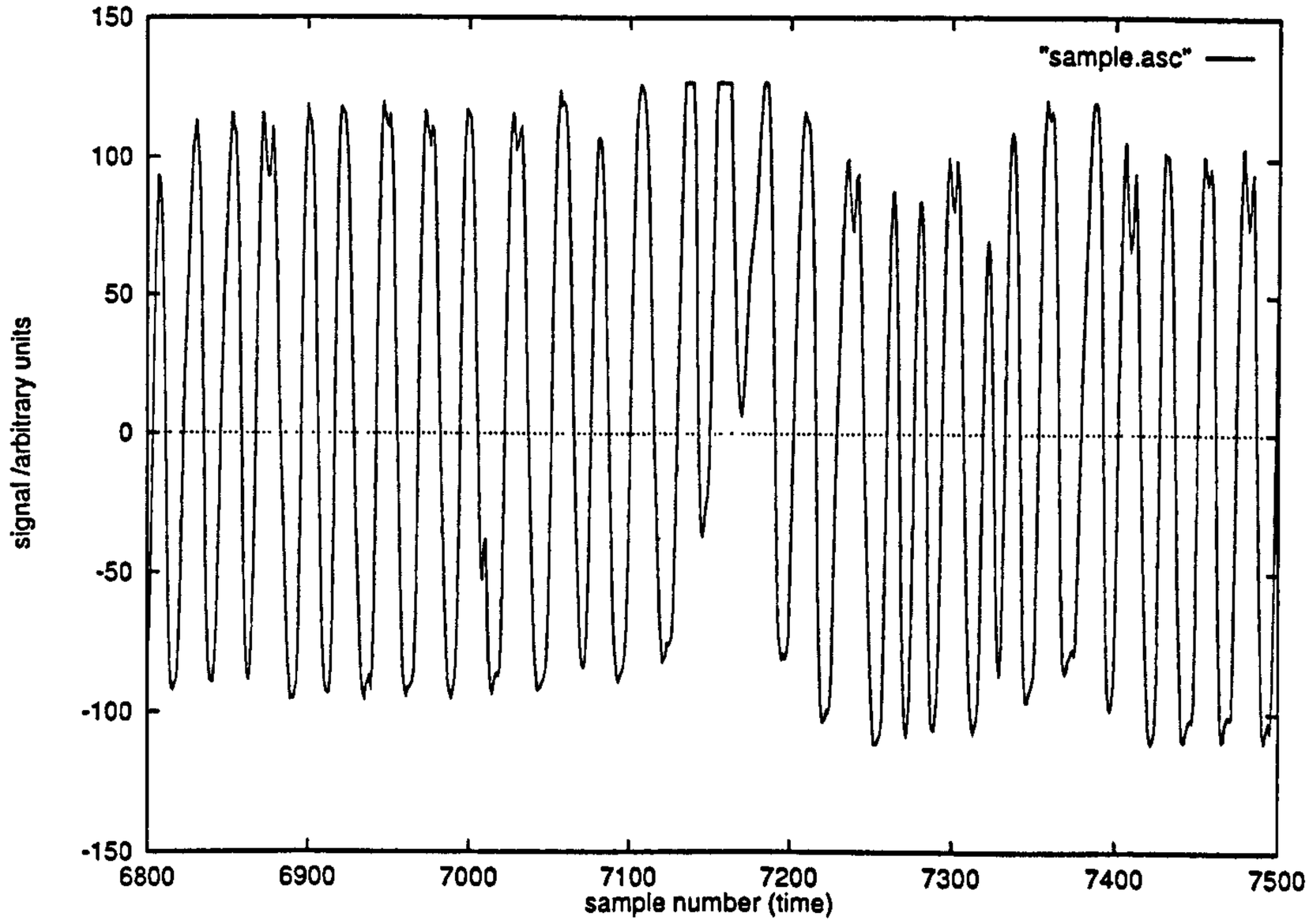


Figure 5.43: Portion of signal for swiped card, showing prevalent MR flip back and a defect.

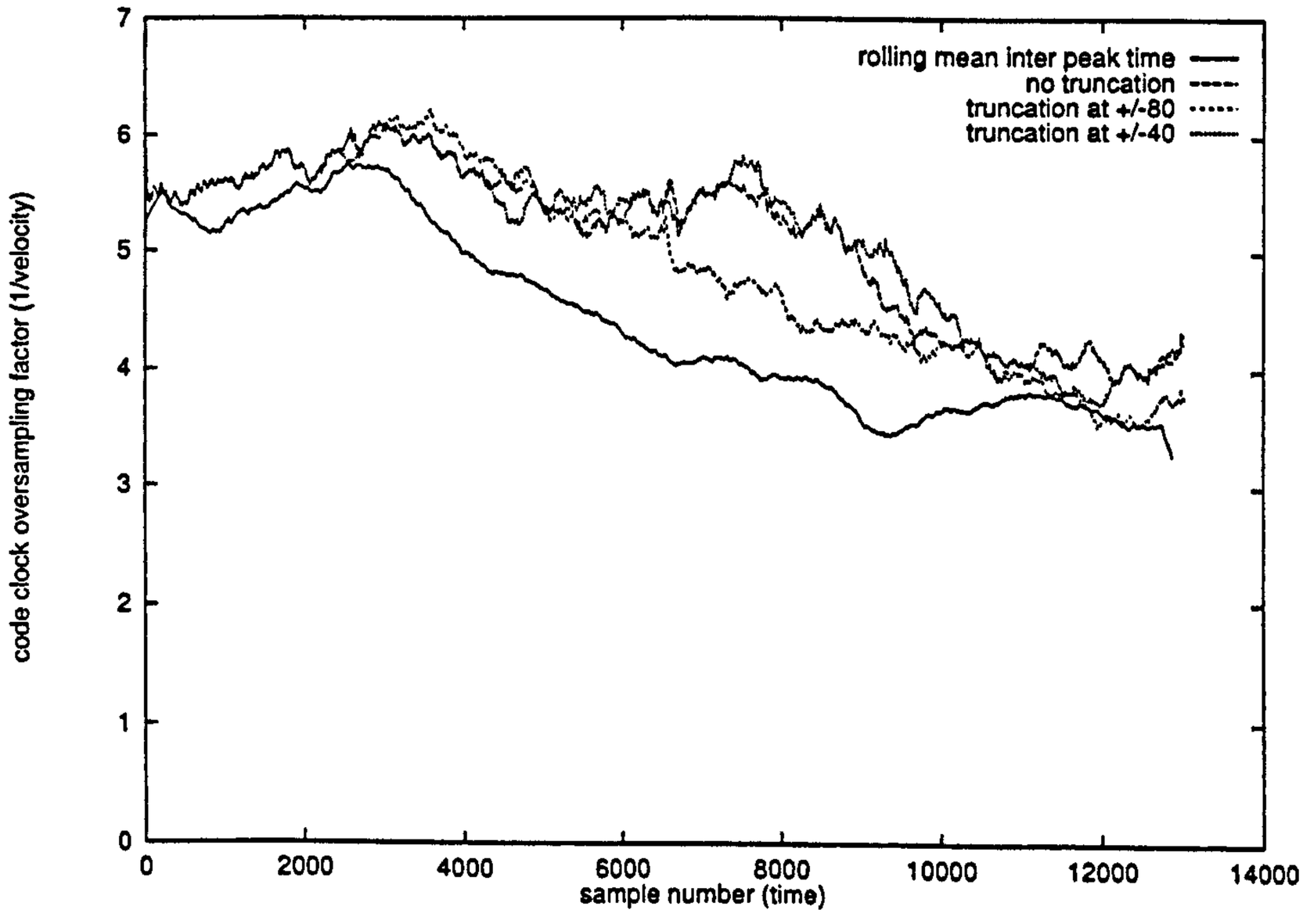


Figure 5.44: Effects of truncation of swipe card signal.

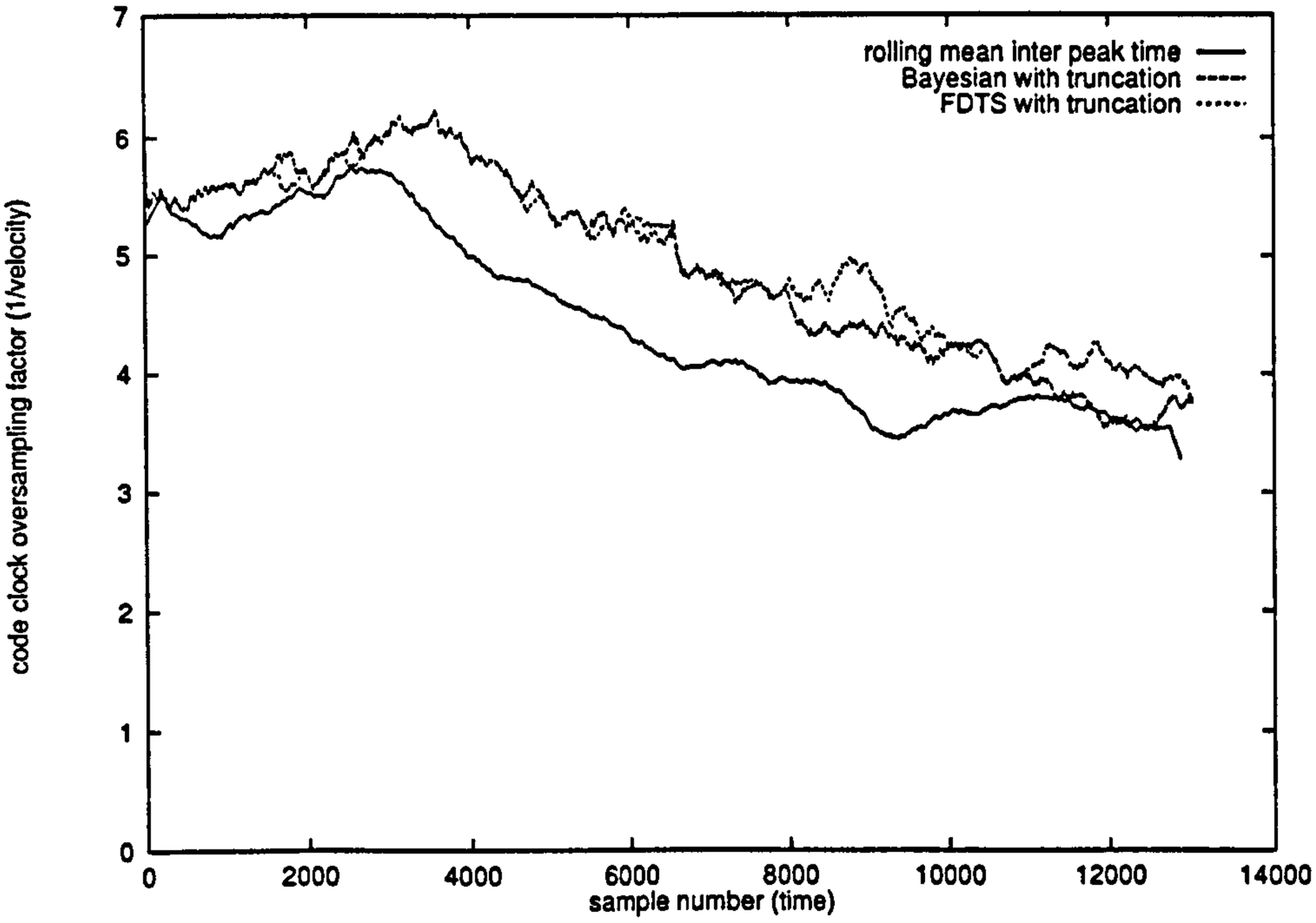


Figure 5.45: FDTS versus Bayesian decoding of a swipe card signal.

and velocity separately. Although this may be appropriate in a steady signal environment, in the swipe card situation any phase correction in fact results in total failure fairly rapidly.

- Fixed delay tree search versus full Bayesian decoding. The difference in computation time is not substantial in this application. The Bayesian treatment should give better results, but this is not certain with real data. Figure 5.45 shows that the difference in tracking ability of FDTS versus Bayesian decoding is not measurably significant.

- Least mean squares adaptive filter feedback. The step size in the feedback is a chosen parameter. This value can also be zero to effectively remove feedback. Indeed, because the initial response was chosen to estimate a 'perfect' channel in the first place, the effect of LMS feedback in figure 5.46 is not enormous. Although feedback improves the confidence margin substantially, it is also the major failure mechanism for very poor signals.

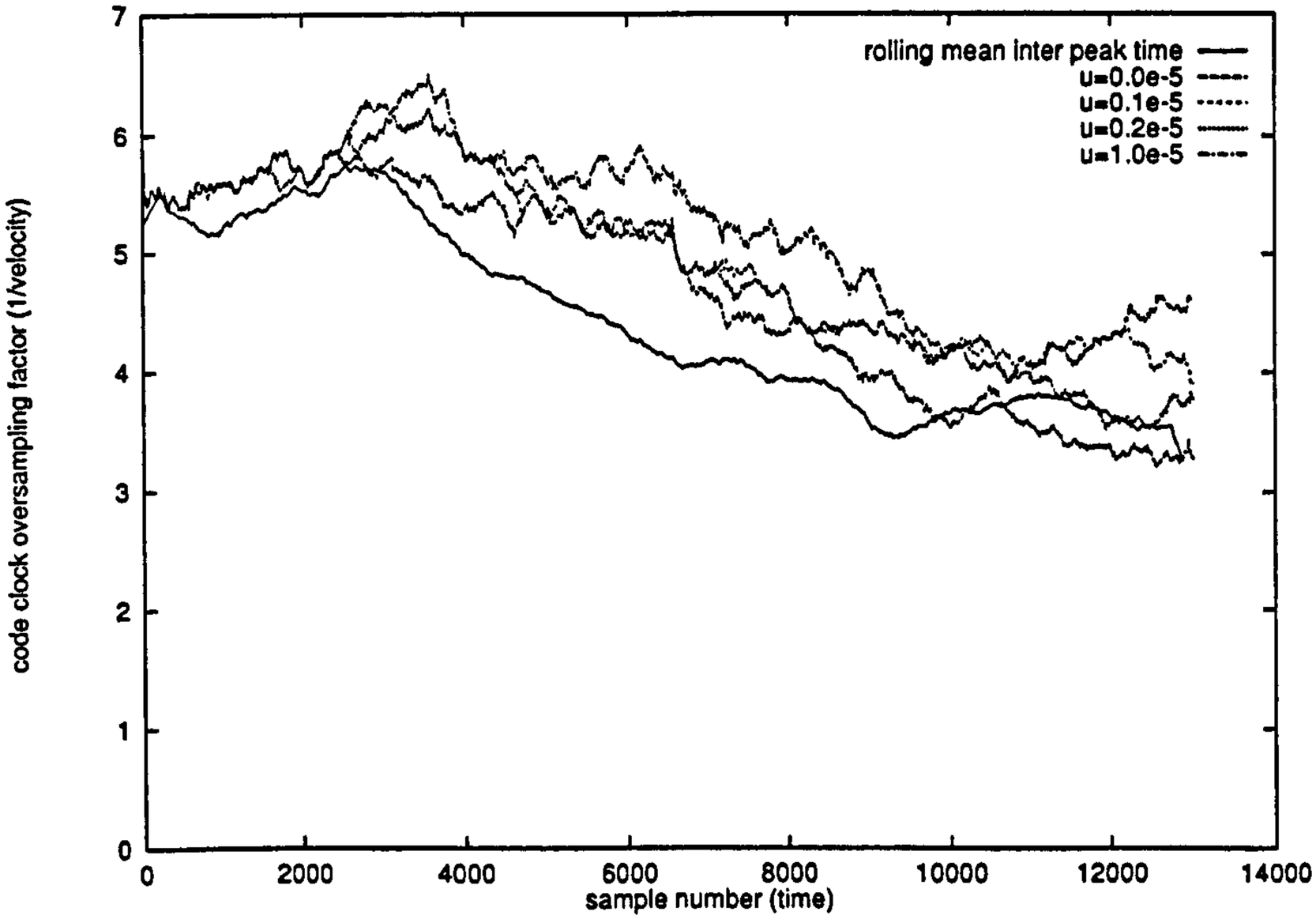


Figure 5.46: Effects of variation of LMS step size for a swipe card signal.

DF: decision feedback

- Decision feedback and look-ahead size. The size of the buffers for look-ahead and DF help determine not just the error rate, but also the timing recovery ability and the adaptive filter size. These are again both choosable parameters. Figure 5.47 shows that the noise and variation in the velocity estimate is considerably higher with a smaller DF buffer size; exactly as expected.

5.7 What makes a 'good' detection scheme?

An optimal detector makes best possible use of the analogue signal data, and it deduces the most probable sequence of digital data that caused it. For a fixed channel environment an arbitrarily optimal detector can be realized. A channel where there are unlimited forms of signal degradation and interference can never have a truly optimal algorithmic detector because it is always possible to

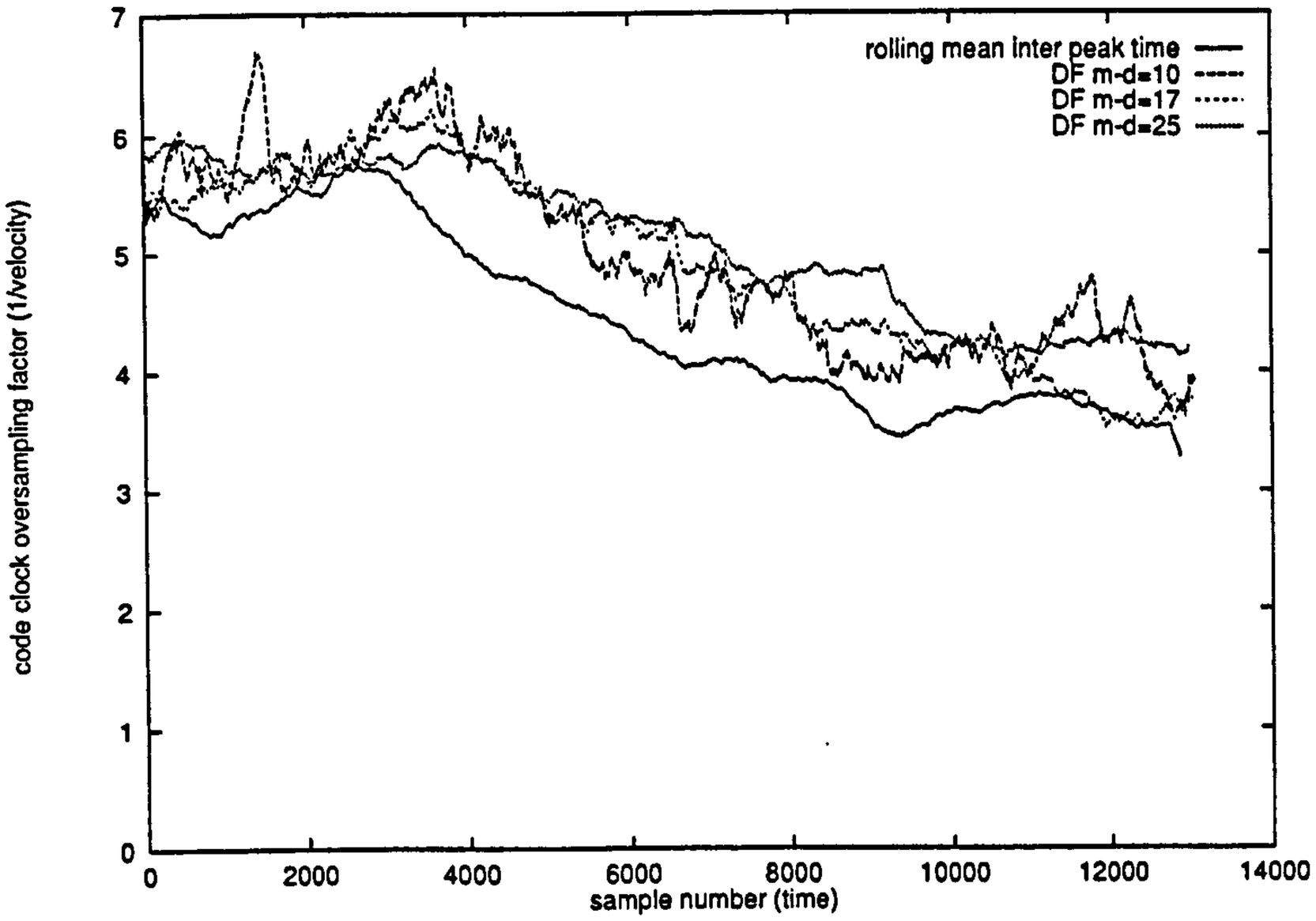


Figure 5.47: Effects of variation of the decision feedback buffer size.

construct a failure mechanism for that algorithm. The credit card environment is such that there are limited forms of distortion and drop-outs, and also a very limited signal period. This precludes the use of a strictly optimal detector, but it is stable enough to allow limited non-optimal detection.

The advantage of a blind detector over a trained detector is a gain in flexibility for the data layout. A training sequence needs to be detected and then used for training—this is area of medium that could be used for storage. However, the blind detector will tend to mis-read the bits that are present while it is adapting to the channel. In a record-then-analyze environment this is not a disadvantage, as is the case with this credit card system. In fact, the largest apparent inferiority of the blind detector is its propensity to adapt incorrectly to the channel. Coupled with the velocity tracking mechanism this has lead to catastrophic data detection failure in some cases.

The collection of consistent results has been hampered by non-systematic, large-extent jitter. This is a fundamental problem that can be solved by chang-

ing the system and medium. However, one of the project intents was to leave the credit card medium unchanged and this constraint severely degrades the effectiveness of the velocity following decoder. Because of their inherent robustness, the discrete peak detectors perform surprisingly well in comparison to the more sophisticated MAP detectors.

MAP: maximum a posteriori

The jitter also leads to the high performance of the inefficient $(d, k) = (0, 1)$ code in comparison to the $(d, k) = (1, 3)$ code, although the variable rate code remains the best choice of efficient code for the medium and detector.

The principal failure mode of the velocity tracking Bayesian detector appears to be unexpectedly large timing errors in the read-back signal itself. Because the velocity offset calculation does not have memory, it applies the result from any erroneous signal immediately. Sometimes this causes code errors and error propagation leading to extensive decoding faults. Another cause is the brief drop-out: normally caused by the head lifting from the card over a defect, see an example in figure 5.43. Again, the velocity tracker sees this as a timing error and acts inappropriately. The solution to both of these problems is to lower the bit density or to frame the data in such a way as to make resynchronization possible. Embedding the data in separated fields or regions (sectors) can potentially help to contain errors, but the severity of the dropouts is determined by the linear recording density (Stevenson loss).

The quality of the swipe action itself can also influence the choice of detector. The swipe affects the velocity, signal amplitude, channel bandwidth, channel consistency and track skew. The Bayesian detectors can be scaled up in complexity to compensate for inter-track interference with a large increase in computation expense (although no attempt was made to try this). The peak detectors can also compensate through linear cancellation. The quality of the swipe can be controlled to a certain extent through the quality of the reader, but mostly through the quality of the card itself. The detection routine must possess a certain roll-over (flywheel) capability to cope with transient signal defects, but it must also be responsive enough to track the rapidly changing natural conditions present in the system. Once again, the simplicity of the peak

detector makes it more robust than the Bayesian detectors. But once the decoding is under way, the more sophisticated detector can pull out a higher user bit density.

In the face of these issues, is there any clearly better detector for the credit card? The reluctant answer is that there is not. For a perfect read every time at minimal cost there is no option but to sacrifice bit density and utilize a peak detector with a highly redundant code. To achieve a higher bit density, it is necessary to move to some form of adaptive system. The Bayesian detector described in this thesis is perhaps the only detector that can efficiently track both velocity and channel simultaneously. However, it is possible that there are superior forms, in terms of bit error rate, that may utilize independent velocity tracking, equalization and detection. For instance, the sliding average zero-crossing detector can be used to produce a form of velocity profile, which can then reform the read-back signal to a constant velocity, which will then be applied to a straightforward equalizer to pass though a Viterbi or peak detector again. However, the accurate determination of velocity is both the most difficult and most critical factor in the credit card detector. A good detector must acquire and follow the card's velocity to achieve any correct bits.

Chapter 6

Conclusions and discussion

This section presents a summary of the main conclusions and results of the investigation. It also briefly discusses some of the issues related to the credit card system as a whole, and some of the steps required to transform the high density credit card from a research project into a tangible article.

6.1 Summary of work

A flexible hardware base was constructed to make a platform that can not only read and write credit cards, but is also capable of a wide range of channel measurements. The card writer was a modified dot-matrix printer, fitted with a laterally adjustable head. The write electronics were digital, with a multi-track parallel write operation at a programmable clock rate. The card reader digitized multiple tracks from a read head and passed the sampled data to a computer. All the data streams were complete in memory before detection and decoding was started.

6.1.1 Coding

Several coding schemes were examined with special attention paid to the efficiency and error resiliance of the codes. Resilience to timing errors was crucial, but becuase this is not a generally common code requirement, the author de-

vised a new code measurement: the timing error resilience (TER). This is a corrected measurement of the mean number of bits in error after a timing and burst data error. Using this measurement it was found that only simple codes (NRZ, FM, MFM) or codes with low efficiency (BPPM) satisfied both the (d, k) constraints of the channel and the TER tests. A new variable rate coding scheme, devised by the author, was also tested and was found to perform exceedingly well in the TER-100 test. This efficient new code class has a rate that is dependent on the data itself and is capable of resynchronizing to the symbol clock on any transition.

6.1.2 Clock recovery

There are many techniques available in the communications area to recover timing phase from data streams. However, they only address the issue of very small timing and phase perturbations. The author devised a phase and velocity offset calculation that relies on the contents of filter buffers as 'snatches' of data. From these it is possible to track and correct a very wide range of velocities. Not only does the snatch offset estimation lock into the frequency in a nearly linear manner, but it is also tolerant of modest non-linear distortions in the system. The snatch estimation requires two sets of signals, both of which are present in a MAP decoder. This method made it possible to decode data streams with no prior knowledge of the speed of the credit card swipe.

MAP: maximum a posteriori

6.1.3 Detection

It was discovered that the channel on a credit card was so variable that no fixed equalizer or partial response scheme was effective. Indeed, a peak detector with no equalization consistently outperformed one with fixed equalization. Since the card's channel would always be initially unknown, and because the entire set of signal information from a swipe was stored in memory, it was determined that a blind MAP detector would be the most effective detection device. A general Bayesian detector was constructed in software, and the snatch velocity

offset calculator was integrated into it. While assessing the probabilities of code sequences, the decoder also factored in the non-linear distortion of the signal attributable to the MR read head.

MR: magneto-resistor

Although the detector performed well with simulated data, decoding successfully at around double the user density of the peak detector. The results were not as good with real signals, achieving only 200 bpi, though still better than a peak detector on this particular type of card. A problem with excessive transition timing jitter in the card signal is the cause. Although the author eliminated many sources of jitter in the system while tracing this problem, the magnitude of the timing offsets still managed to defeat the snatch offset calculation at medium bit densities. Once velocity tracking was disturbed, there was no chance for correct decoding.

bpi: bits per inch

It is believed that the timing jitter is fundamental to the credit card magnetic medium. To proceed further, avenues of research into both physical causes and algorithmic correction of jitter are required. It may require the magnetic stripe card medium to be changed before a high density credit card can be realized that can compatibly store 3 kBytes of information.

6.2 The high density magnetic credit card system

To an observer, a high density credit card system will work like this in a retail environment. After the bill has been finalized, the (potentially blank) card will be swiped (or inserted) into the reader and returned. A signature chit will be printed and the photograph of the card-holder will appear on a screen. Once the signature has been taken, the signature as retrieved from the card will be displayed on the screen for comparison. Online authorization from the card issuing company is an option that the vendor can still exercise, in addition to existing fingerprinting and watermarking card security options.

Internally the only new feature of the system is the card reader and the display system. The display is straightforward and a small colour liquid crystal display will not add an enormous amount to the cost of a point of sale termi-

nal. The physical reader, if a swipe mechanism is used, is a significant expense, probably running at 2–10% of the cost of the EPOS unit. It is not currently known whether the accuracy advantage of a mechanical reader over a swipe reader is cost and time effective. The detection and computer processing facilities are already sufficient in most terminals to support the system.

EPOS: electronic point of sale

But given the current low level of credit card fraud against the extra cost of the reader, the system would only find limited use certain high-risk retailing establishments. Even the act of comparing signatures with the current credit card procedure is of economically indeterminate value in many situations.

6.2.1 Reliability issues in the field

The high density credit card will undoubtedly have a higher raw bit error rate than a standard card. Whether this rate is still low enough for error correction to eliminate is uncertain: the dominant field failure mode may be such that error correction is not feasible, for example if the card skew is prominent or if the accumulation of debris in the reader and on cards enlarge the head-medium spacing overly. Even with the standard magnetic cards there is a distinction between the credit cards, which operate at 75 and 220 bpi, and identification and token cards, that operate at still lower densities, often 40 bpi on a single wide track. This latter class of swipe cards is designed to be used rapidly and by untrained people, and must often suffer much worse abuse and reading conditions than credit cards. The pertinent data on the card is usually repeated*, so that even a partial swipe will be valid. With framing taken into account, the authentication procedure on these cards operates with only two or three bytes of data. This is sufficient in many cases: low risk conditional access, employee number, and so on, but is probably insufficient from a security standpoint with regard to token payment systems. However, the token cards are usually designed to be rewritable, and cannot use a higher density because of the reliability versus mechanism cost argument.

bpi: bits per inch

*One interesting exploitable consequence of this is that it may be possible to cut the card in half—each half still working—making duplication trivial.

The failure mechanisms for identification, token and credit cards are different. The author has seen identification cards folded, defaced, peeling and warped, but token and credit cards are given more respect and are normally carried in a wallet or purse. The failure mechanism here is often cracking along the track, medium wear, demagnetization and foreign body deposits. The failures are often catastrophic but seldom occur.

Soft failure—the failure to read a card on the first but succeed on a subsequent attempt—is usually down to the orientation of the card through the reader. Cleaning the card is often of secondary importance to moving the card steadily and deliberately through the slot.

All this suggests that for a high density card to be useful in the field, it must be very accommodating towards misregistration. The layout described in section 2.4.5 allows for up to 0.6 mm of drift with an eight-track read head. The credit card standard, however, allows for almost 5 mm and yet misreads still occur. Subsequently, for the narrow track pitch scheme to work transparently, more redundant heads will be needed across the stripe. The other solutions to this are operator training (which is fairly minimal) or a fully mechanical reader. All three solutions involve extra expense at the read point.

6.2.2 What *is* the most important property?

The bottom line is economics. Although the general trend in the magnetic recording industry has been to reduce the cost-per-bit of data storage, the driving force in credit card manufacture is to reduce the cost-per-card. Consequently, the current form of plastic swipe card is such that its data storage capacity is only slightly in excess of that required by the credit card standards, and high density storage is not a natural proposition on such a card. This study has not addressed the real-world issue of cost and the actual use of a high density credit card system will only be investigated commercially if the following points are determined to be economically favourable overall.

- The extra cost of the card and its manufacture.

- The extra cost of the reader and associated machinery.
- The added cost of new card failures.
- The added cost of operator training.
- The difference in transaction processing time.
- The reduction in fraud due to increased card capacity.

A Smart card system has all of the above features, with the exceptions that the reader unit is less expensive and the card cost is significantly higher. However, the volume shipping of Smart cards may make the increased card cost marginal. There are already moves to integrate electronic cash and credit cards onto one general purpose payment card. It will take years for this new fiscal form to establish itself, and the high density credit card has nothing except familiarity with which to promote itself for a long-term position in public financial transactions. However, the outlook is perhaps not as bleak in other areas. For niche applications requiring hardy and inexpensive cards, the high density magnetic card may find a market. For example, a social or medical card: where there are many cards, few readers and few transactions.

6.3 Further work

The high density magnetic credit card research is not complete and encapsulated. It is quite clear that the work in velocity tracking can be improved upon. Once a bit has been decoded in error, the rapid feedback from the velocity update mechanism can severely disrupt subsequent bits, propagating the error. However, despite this the method in itself is remarkably stable, even in the face of near 50% bit error rates. The snatch velocity estimation technique has been applied to comparatively short bursts of data: its behavior over long term variable speed signals has not been studied. In addition, the behavior of the method with consistent channels (i.e., a channel where the conditions change over a few

hundreds or thousands of bits, rather than on the scale of one or two bits) has not been investigated.

A multiple iteration method is also ripe for study. This involves scanning the signal to pick out statistics then running the detector over the signal using the statistics from both forwards and backwards in time. For example, a simple peak detector could be used to pick out special synchronization symbols spaced every hundred code clock units, and from their separation determine the velocity profile of the segments. As long as the velocity does not change rapidly, a conventional detector could interpolate linear or parabolic speed changes to pick out the data. This requires accurate detection and positioning of the sync patterns, which is not trivial. Another method, previously mentioned, is to run the variable speed Bayesian detector over the signal backwards to adapt to the channel, then forwards to decode the data. However, preliminary testing of this idea was disappointing, probably due to the very high initial error rates.

Another aspect that has not been examined is the extraction of interfering track signals. This again may require the sectoring of data on the stripes to insulate one data block from the next, and protect each block with an error detecting code. Then every block from every head can be decoded, and the matrix of correct blocks can be used to recover the data. An intriguing possibility is the expansion of the Bayesian method to include not only the signal contributions from the primary track, but also contributions from a neighbouring track. This squares the computation time, and has not been investigated further for that reason. Increasing the sophistication of the Bayesian detector regarding non-linear distortion does not look rewarding at this time.

Probably the most fruitful area of research for a high density magnetic card is the card itself. The answers to the question of why there is so much magnetic jitter may reveal a solution to higher recording densities. However this investigation would be a complete project in itself.

6.4 Summary

It has been shown that in principle it is possible to store 20 kBytes within the boundaries of track 3 of a standard magnetic credit card. However, the practicalities of coding, medium granularity leading to non-systematic jitter and velocity following in the face of bit errors and signal drop-outs due to the high density recording, the limit of accurately detectable coding is about 300 Bytes within track 3 (equating to 200 bpi on four independent tracks with ECC overhead). This is still a four-fold increase over standard density recording, but is disappointing nonetheless. Inaccurate decoding can push the figure closer to 1500 Bytes in track 3. A solution to the jitter problem would make this figure easy to achieve.

ECC: error correcting code

Appendix A

Published papers

These papers, relevant to the credit card project, were submitted and published in the following journals (writing date order) and are reproduced here.

- D.J. Mapps, N. Fry and D. Smith.
  Non-linear magnetoresistance.
  Studies in Applied Electromagnetics and Mechanics. pp 628–633
  IOS Press, 1996. ISSN 1383–7281.

- D.F. Smith, T. Donnelly and D.J. Mapps.
  High density storage on a magnetic stripe card.
  IEEE Transactions on Magnetics. Volume 32(5), pp 4025–4027,
  September 1996.
  Presented at the Intermag Conference in Seattle, WA, USA in April 1996.

- D.F. Smith, T. Donnelly and D.J. Mapps.
  Fixed sample rate Bayesian detector in a variable speed magnetic channel.
  IEEE Transactions on Magnetics. Volume 33(5), pp 2797–2799,
  September 1997.
  Presented at the Intermag Conference in New Orleans, LA, USA in April 1997.

Nonlinear Electromagnetic Systems
A.J. Moses and A. Basak (Eds.)
IOS Press, 1996

Non Linear Magnetoresistance

D J Mapps, N Fry and D Smith
Centre for Research in Information Storage Technology
School of Electronic, Communication and Electrical Engineering
University of Plymouth, UK

Abstract. This paper discusses the phenomenon of magnetoresistance in thin magnetic films in terms of its use in sensors and magnetic replay heads for computer hard disk systems. Theory is developed which indicates that non-linearity is a fundamental problem in these detectors. Various methods for coping with non-linearity are described as well as some ideas for defining the upper and lower limits of the linear operating region in practical devices. Some problems in the future use of magnetoresistors having 'giant' magnetoresistance (GMR) are highlighted.

1. Introduction

The phenomenon of magnetoresistance was discovered more than one hundred years ago but it is only in recent times that it has emerged as an important effect of use in magnetic

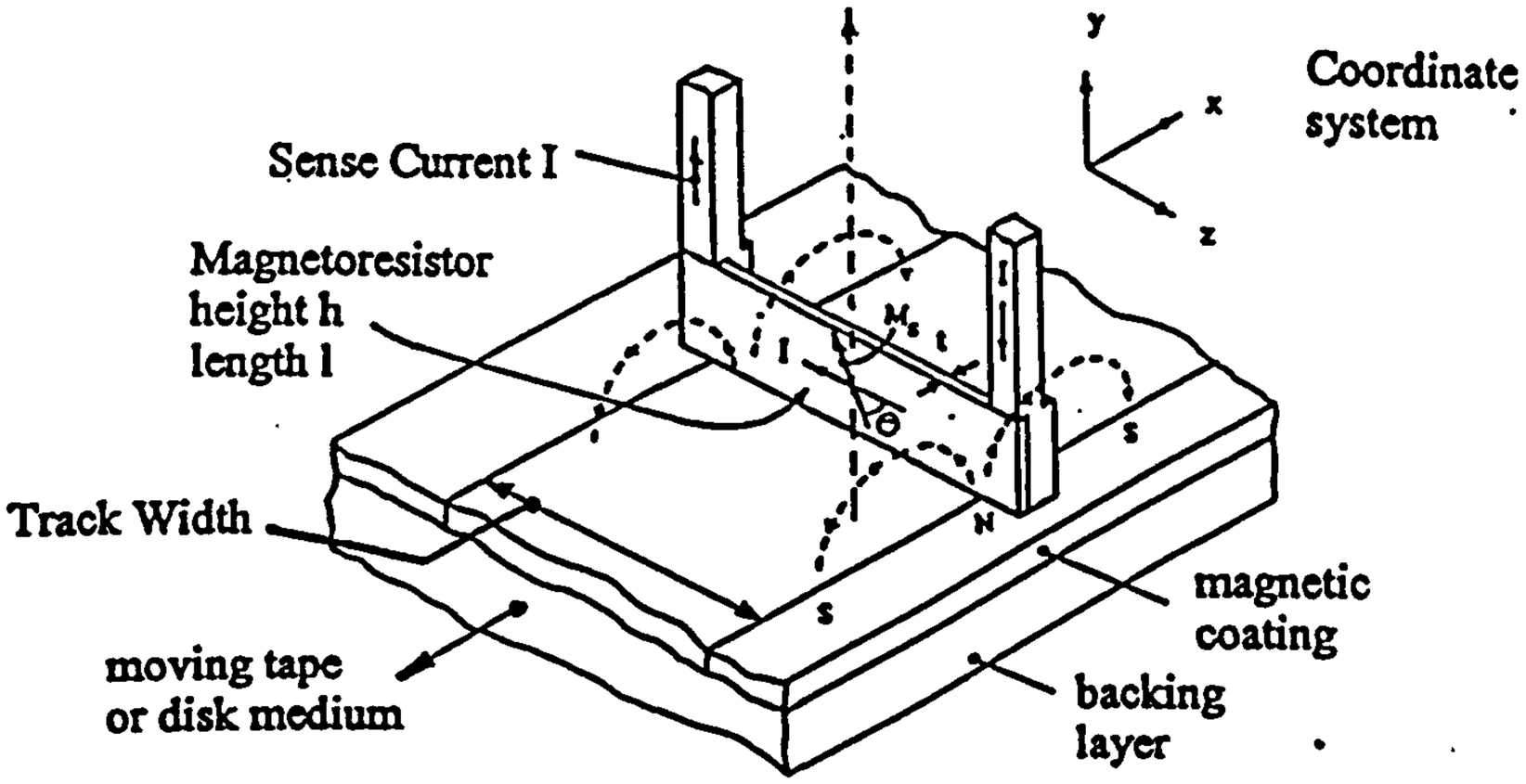


Figure 1. Playback using the magnetoresistance (MR) effect.

field detectors. The basic effect is found in magnetic thin films, usually less than 100 nm thick where an externally applied magnetic field can change the natural state of magnetisation of a thin-film and thereby affects its electrical resistance. Such a change can be used in a replay head in a magnetic recording system to detect information on a tape or disk (see figure 1 from reference 1) or in a sensor to detect magnetic fields in the micro-Tesla regions (ref. 2). The ability of such thin-film devices to be photolithographically patterned means that they can be made with micron dimensions, so leading to great advances in recording resolution on computer disks. This has enabled disk

information storage densities to rise at the rate of about 60%/annum (see ref. 3). The great potential of these devices as magnetic field detectors has yet to be developed but it is clear that their small size, low power and high sensitivity will make them potentially strong competitors for applications currently provided for by fluxgates, coils and Hall probe sensors.

2. Theory

Figure 2 shows a rectangular-shaped thin-film magnetoresistor in a single domain state. The magnetisation vector M_s makes an angle θ with the long axis of the film and the film magnetocrystalline anisotropy is at an angle γ. An applied field H_R is in the plane of the

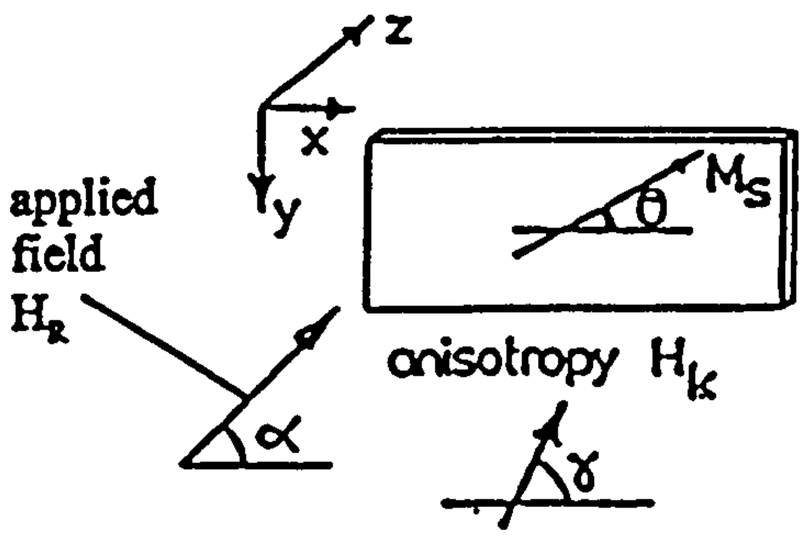


Figure 2. Showing the field, anisotropy and magnetisation directions

film and makes an angle of α with the long axis, along which a sense current I is allowed to flow. The system can be analysed by considering the various magnetic energy components that exist in the film i.e.:-

(i) The applied field exerts a twisting force on the magnetisation in the film and sets up a potential energy given by,

$$- M_s H_R \text{Cos} (\alpha - \theta) \qquad (1)$$

(ii) Anisotropy energy resulting from the magnetisation vector being twisted away from the state of low magnetocrystalline energy,

$$+ \tfrac{1}{2} M_s H_K \text{Sin}^2 (\gamma - \theta) \qquad (2)$$

(iii) Demagnetising energy in the film due to the free poles at the edges is;

$$+ \tfrac{1}{2} M_s^2 N_D \text{Sin}^2 \theta \qquad (3)$$

If the energy components of (i), (ii) and (iii) are summed and differentiated with respect to θ, the resulting expression can be equated to zero for an energy minimum representing a stable magnetisation in the film.

If M_s is in the plane of the film the demagnetising field through the film thickness can be discounted. For a film with reduced height compared with its length, only the transverse demagnetising factor (Ny) need be considered. Hence, summing and differentiating gives,

$$H_R \text{ Sin } (\alpha - \theta) = \tfrac{1}{2} H_K \text{ Sin } 2(\gamma - \theta) + \tfrac{1}{2} N_y M_S \text{ Sin } 2\theta \qquad (4)$$

from which a graph of H_R against θ can be plotted, assuming α, θ, γ and the film constants, are known.

In magnetic recording applications, the field from the tape or disk is usually along the y - axis of the film (i.e. $\theta = 90°$) and the anisotropy axis is selected to be along the x axis (see figure 1) making the angle γ = zero.

Equation (4) simplifies to,

$$\text{Sin } \theta = \frac{H_y}{H_k + N_y M_s} \qquad (5)$$

The relationship between magnetoresistance $\Delta\rho$ and the angle θ for the case of the sensor shown in figure 1 is,

$$\rho = \rho_o + \Delta\rho \, Cos^2\theta \qquad (6)$$

Combining equations (5) and (6) leads to fractional magnetoreistance as,

$$\frac{\rho - \rho_o}{\Delta\rho} = \left[1 - \left(\frac{Hy}{H_k + N_y M_s} \right)^2 \right] \qquad (7)$$

If H_y is made up of a constant component, H_b and a much smaller variable component h_y then, for a current I and initial resistance R_o an output voltage can be obtained by integrating over the device dimensions, i.e.

$$v = 2IR_o \cdot \frac{\Delta\rho}{\rho_o} \cdot \frac{H_b}{[H_k + NyMs]^2} \iint h_y(y,z) \frac{dy}{h} \cdot \frac{dz}{l} vo \qquad (8)$$

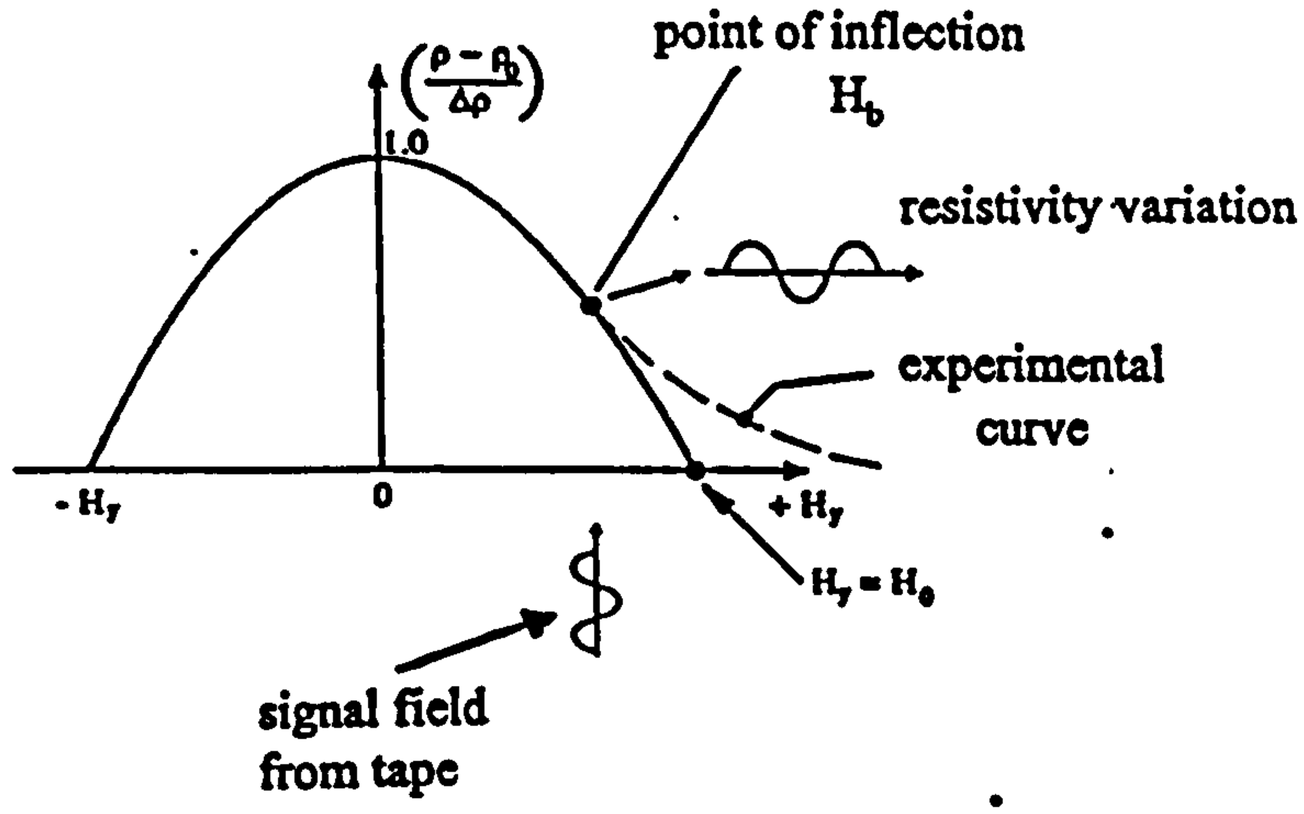


Figure 3. Theoretical curve based on the relationship of equation 7 and the corresponding experimental curve related to magnetic saturation effects.

The significance of the constant component of field H_b can be seen if the fractional magnetoresistance change is plotted against transverse field H_y as shown in figure 3. As can be seen the theoretical curve has no linear regions but in practice, saturation effects

cause the curve to be asymptotic to the field axis, creating a point which enables the thin-film magnetoresistor to be used as a linear device. The extent of the linearity depends on the way the material saturates and this depends on a number of materials and dimension-related factors. The sensor is usually placed in a d.c. bias field H_b and subjected to an alternating signal field h_y (see figure 3).

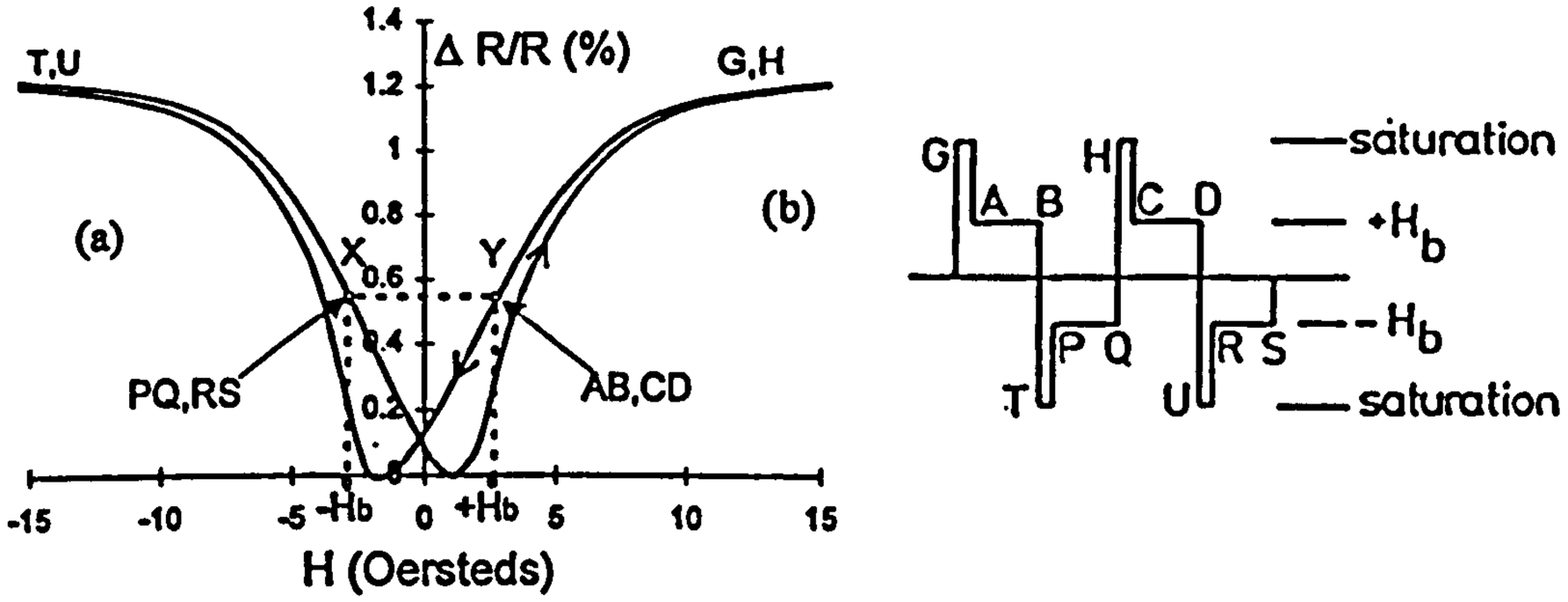


Figure 4. (a) Typical magnetoresistance curve showing hysteresis. There are two possible values of $\Delta R/R$ for each value of H_b. Preferred points of operation in a differential system can be established by an active bias field of the form shown in (b).

3. Assessing Linearity

An actual magnetoresistance curve is shown (inverted) in figure 4. As can be seen, because of hysteresis, there is not a curve, but a loop, and, consequently, there are two values of magnetoresistance for every value of applied field except when the sensor saturates. Just as for a B-H loop in magnetic materials, there is only one direction around which the loop is traversed (shown by the arrows in figure 4). The linearity of falling and rising curves is, generally, different. Also, micro discontinuities in the sensor arising from physical or materials imperfections cause the curve to be non-smooth in practice and this in itself produces troublesome non-linearity which must be eliminated.

Linearity may be defined in terms of any operator L which operates on a function x to produce another function y (i.e. $L(x_1) = y_1$. If L is linear then the following equations hold:

$$L(x_1 + x_2) = y_1 + y_2 \tag{9}$$

and

$$L(k x_1) = k y_1 \text{ where k is a constant} \tag{10}$$

The meaning of equation (9) in the context of a transfer characteristic such as in figure 3 is that if another signal x_2 is added to the first then the net effect on the output is that the signal y_2 would have a simple addition to x_2 i.e. like a kind of direct superposition.

The meaning of equation (10) is that if x_1 is multiplied by a constant k and the system (operator) is linear then the result is that the output y_1 is multiplied by the same constant k, i.e. k is like the slope of the MR transfer curve near the point of inflection H_b.

In practical terms, the linear portion of the curve will be exceeded if the added signal x_2 extends the overall signal to too large a value. This also applies if the multiple k is too great.

One of the best ways to explore linearity of MR curves is by differentiation. This is shown in figure 5 which shows the first and second derivatives of a typical half-curve. The region where the first derivative reaches a maximum is the place of maximum sensitivity and, possibly, greatest linearity. A clearer indication is given in the second derivative by the point of zero crossover and the magnitude of the ordinate values near this point. A further method is to divide the second derivative (Q) by the square of the first derivative (P) plotting the square of this against H, $[Q/P^2]^2$. This has the advantage of amplifying the 'linear' effect as shown in figure 6 which compares the result with the original MR curve. The difference in upward (a) and downward (b) parts of the magnetoresistance hysteresis curve based on this method are illustrated by a wider flat region in the ratio-ed, squared curve for (b) which is inherently better for use in a field detector transducer.

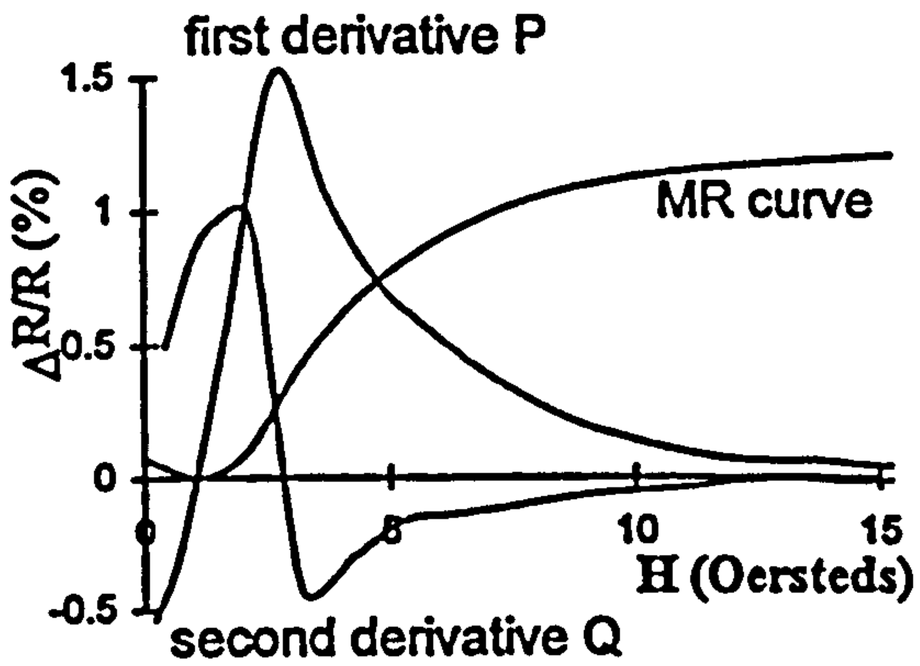


Figure 5. Showing a half magnetoresistance curve with its first and second derivative.

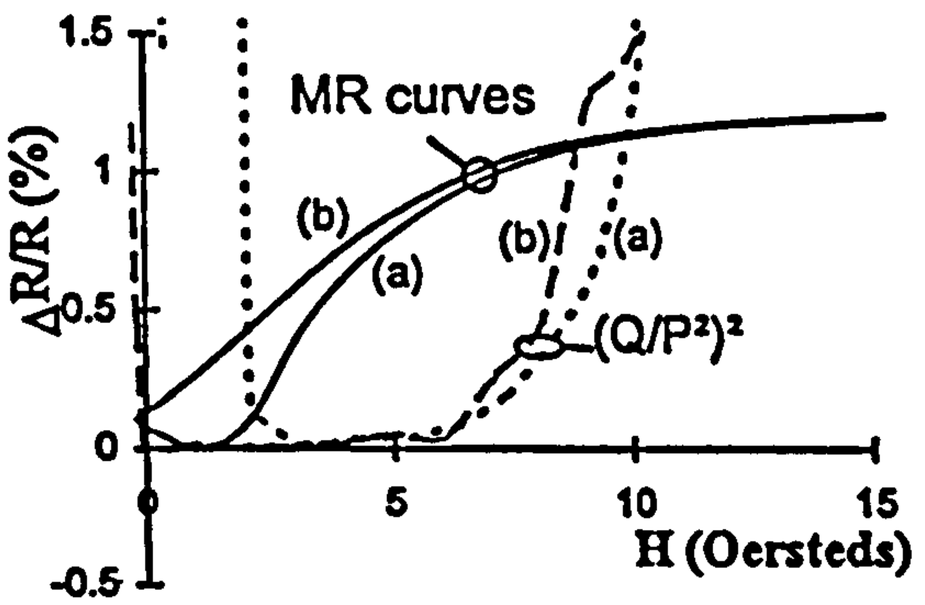


Figure 6. Comparing the linearity of (a) falling and (b) rising parts of a magnetoresistive curve subject to hysteresis.

4. Practical Solutions to the Non-linearity Problem

A practical way of improving linearity is use a system with two MR elements biased in opposite directions with their outputs fed into a differential amplifier. This method is used in reference 4. The differential system essentially subtracts out the unwanted harmonics from the signal and produces a linear response. In practice this does not remove the anomaly shown in figure 4 when hysteresis is present. It can be seen in figure 6, however that if the bias point used is the one on the curve immediately after transverse sensor saturation then linearity is obtained over a larger part of the curve than for the point before sensor saturation. For such points identified in figure 4 (a) as X and Y, an active bias field can be employed such as shown in figure 4(b). The sensor is driven into saturation at G, and the measurement made immediately afterwards at the H_b level identified by AB, so ensuring that there is no anomaly in determining the correct operating point and the sensor operates at optimum performance.

Many other practical solutions have been developed for simple anisotropic magnetoresistance sensors as detailed in reference 1. Recently, however, a new class of magnetoresistors is emerging which consist of a number of antiferromagnetically coupled

layers half of which can be switched by the application of a signal field. If optimised carefully, these 'giant' magnetoresistance (GMR) sensors can be made to change coupling with comparatively small signal fields resulting in large magnetoresistance change several times bigger than previously known. One of the drawbacks of such devices at the moment is that the effect can be very non-linear and subject to hysteresis. An example of a GMR spin-valve type is shown in figure 7(i). A better system may be the multilayer of figure 7(ii) which has flatter ranges. Unfortunately, it is the spin-valve type which may be preferred in practice, because it is cheaper to manufacture giving system designers great challenges to achieve a linear response.

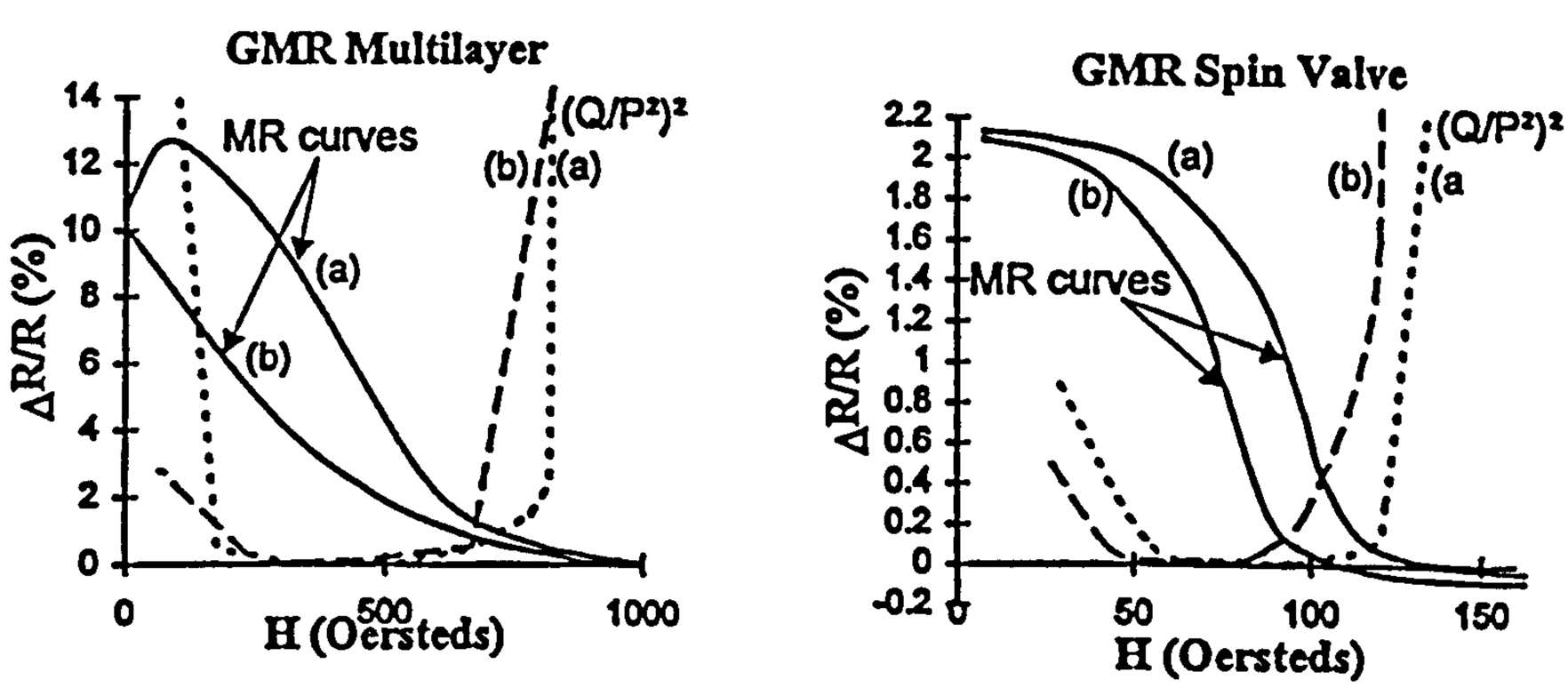


Figure 7. Comparison of linearity of GMR materials (i) multilayer (ref. 5) and (ii) spin-valve (ref.6).

5. Conclusions

Analysis of the linearity of magnetoresistive sensors shows that a differentiation method can be a way of comparing their performance. Using this method it is clear that, when hysteresis is present it is advisable to operate on the part of the MR curve immediately after saturation. Such operation can be achieved using a dynamic biasing system.

'Giant' magnetoresistance sensors present new problems of non-linearity which can be analysed using the differentiation technique.

References

[1] D. J. Mapps, Chapter on Magnetoresistance. Applied Magnetism. ISBN 0-7923-2622-9. Kluwer Academic Publishers, 1994.

[2] D. J. Mapps, M. L. Watson ,and N. Fry, A Double Bifilar Magneto-Resistor for Earth's Field Detection. I.E.E.E. Transactions on Magnetics, Vol. Mag. 23, No.5, Sept. 1987, pp. 2413-2415.

[3] E. Grachowski and D.A. Thompson, Outlook for Maintaining Areal Density Growth in Magnetic Recording. I.E.E.E. Transactions on Magnetics, Vol. Mag. 30, No.6, Nov. 1994, pp. 3797-3800.

[4] D.J. Mapps, U.K. Patent No. 2202635, Nov. 1991.

[5] T. Valet, D.J. Mapps and T. Troup. [$Ni_{80}Fe_{20}$Cu,Co,Cu] multilayers: Candidate materials for readout head applications. BRITE EURAM, PROJECT BE4112, July 1995.

[6] B Dieny et. al. Giant magnetoresistance in soft ferromagnetic multilayers. Phys. Rev B., Vol. 43, No.1, Jan. 1991.

TEXT BOUND INTO

THE SPINE

High Density Storage on a Magnetic Stripe Card

Daniel F. Smith and T. Donnelly and D. J. Mapps
Centre for Research in Information Storage Technology, School of Electronic, Communication and Electrical Engineering,
University of Plymouth, Drake Circus, Plymouth, PL4 8AA, UK

***Abstract*—A new variable rate code with soft-decision decoding has yielded a storage capacity of 2 kBytes (30 kbit/in^2) on a standard banking credit card. The capacity can be further increased through higher track density and using a computationally intensive Bayesian blind inverse filter. This work is continuing to an estimated 4 kByte (100 kbit/in^2) capacity while remaining compatible with current systems.**

I. Introduction

A standard credit card stores data on 3 tracks at 210, 75 and 210 bits/in (1 kbit/in^2) with a self-clocking code (FM) [1]. This gives 140 Bytes of storage, of which only 20 Bytes (track 2) are normally used. This paper describes a high density magnetic storage system on a standard credit card.

A. The Problems

From a signal processing standpoint, a stripe card differs from a disk or tape in two major ways.

1) Channel characteristics: the bandwidth, signal amplitude, etc. are unknown before every swipe.

2) Speed: the velocity can vary considerably.

Both factors can also change during the swipe. (See fig. 1.)

B. Possible Solutions

1) The Smart Card: this has no hard limits on storage capacity and can also be made very secure. However the cost is an order of magnitude greater than that of a plastic card.

2) High density credit card: our approach has been to take a standard credit card and inexpensive, robust hardware then find the card's storage limit. This has been done using multiple tracks, a new storage code and modern, highly processor intensive, detection techniques.

C. The Benefits of Capacity

A digitized signature can be stored in 1 kByte or less (fig. 2). A passport-quality photograph can be stored in around 2 kBytes. (Kodak [2] has compressed faces into 50 bytes—with significant loss of quality—for storage on credit cards.) By removing the card-owner's signature and face from the surface of the card so that they are only accessible electronically, fraud becomes much more difficult.

Manuscript received February 29, 1996. Revised May 1, 1996.
D. F. Smith, e-mail dfsmith@cr1.see.plym.ac.uk, phone +44 1752 232529, fax +44 1752 232583.
This work is being financed by the University of Plymouth.

II. System Description

This high density card achieves gains through higher track and linear densities. There are three main components.

A. Card Reader and Writer

1) Reader: development was performed with a 4-track inductive read head, but the system has been developed so that cards are now hand swiped past an 8-track MR read head (from a Philips Digital Compact Cassette player). The replayed track signals are stored, in their entirety, on computer for processing. This constitutes the hardware requirements of an EPOS (electronic point of sale) machine.

2) Writer: the writing system relies on multiple passes of a large-gap conventional credit card writing head. Twelve tracks are laid down at 130 tpi (195 μm pitch) so that they fit within track 3 of the ISO standard layout. The edge tracks are duplicated to allow for lateral card drift, i.e., eight different tracks are recorded, in the order 7 8 1 2 3 4 5 6 7 8 1 2.

B. Coding Scheme

Tape and disk systems often have a constant bit rate or density requirement. However, in a write-once system with no sector restrictions a variable rate code can be used.

A new self-clocking code was devised for the system (fig. 3). A group of three bits is represented by exactly 2 transitions. The scheme achieves a density ratio of about 1.0 with a (d, k) of $(1, 3)$. The most common 3-bit combinations are assigned the shortest symbols, improving the storage density. Also, synchronization sequences are placed at known intervals along the track to help with speed determination.

In this application, the code has a number of advantages over other density ratio 1 codes.

1) It is hard to misplace bits: to lose or gain 3 bits, two transitions must be lost. Nevertheless, with a large dropout (perhaps due to a surface defect) it can be hard to know how many symbols were lost. Synchronization symbols inserted

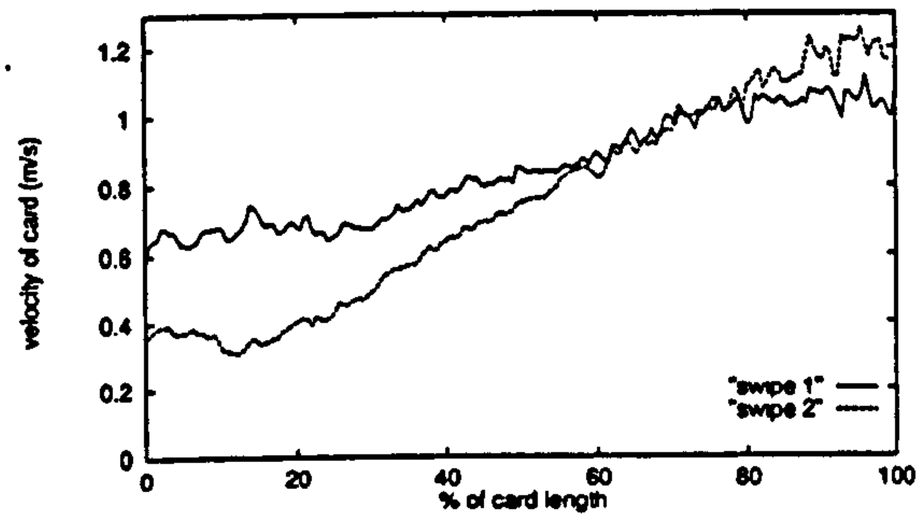


Fig. 1. Example of velocity variations during a swipe through a hand reader.

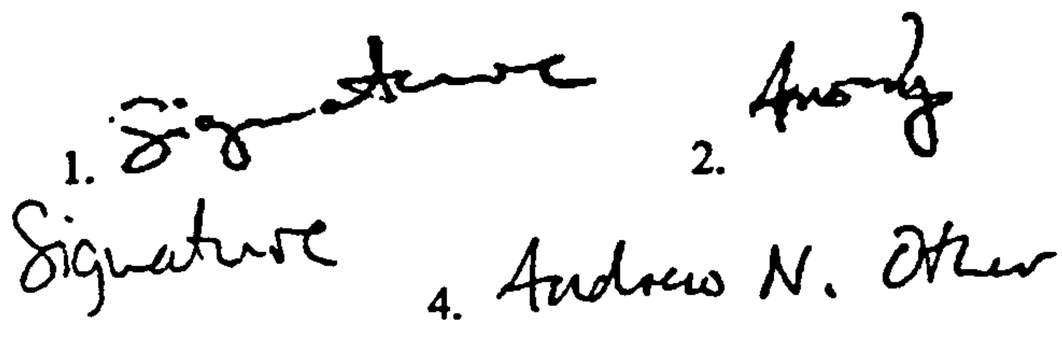


Fig. 2. Example digitized signatures, $\frac{1}{2}$ kByte (1 & 2), 1 kByte (3 & 4). (Simple bitmap compression.)

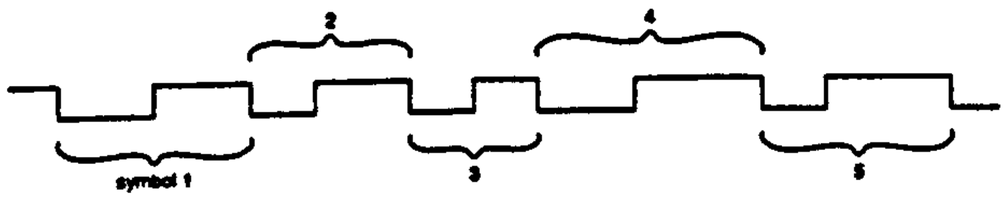


Fig. 3. Variable rate code. Transitions are spaced by 2, 3 or 4 intervals. A symbol is formed by 2 transitions, giving 9 combinations. Eight of these symbols are designated for data (3 bits) and the ninth is used for synchronization.

into the stream allow the number and location of dropouts to be gauged.

2) Every symbol begins with a high to low transition: this makes clocking easier than with standard codes and always allows the decoder to find the start of a symbol, regardless of the preceding symbols.

C. Signal Detection

The readback signal can suffer dropouts, amplitude variations and speed variations as well as inter-track interference. This means that the detection process has to determine the channel characteristics before decoding the signal or use a method that is robust enough to ride out the variations. All the signals are stored digitally on computer and processed off-line.

For moderate density recording and reading with an inductive head, peak detection with no pre-filtering has proved more reliable than other straightforward detection systems. Simple non-optimal pre-filtering was only marginally beneficial.

For higher densities, an MR head has been used, and investigations are ongoing into computationally intensive, blind inverse detection techniques (section III.).

D. Incidental Considerations

This project aims to provide a high density data transport medium using inexpensive equipment with a low error rate. A commercial system, however, must be completely reliable, hence an error correcting code mechanism is needed for the data.

Also, to prevent people from writing their own signatures and pictures onto the card, a public key cryptosystem would make alterations 'impossible.'

III. Inverse Filter Theory

A. Blind Equalization

The channel response of a swipe card cannot be predetermined for a hand reader. There is wide variation in both the card parameters (thickness, medium coercivity, roughness, warping) and the reader parameters (speed, azimuth, lateral displacement). This means that any equalization system cannot be fixed at manufacture, but must adapt for each read. With no training symbols, the equalizer is said to be blind (i.e., both the channel and data are unknown).

A blind equalizer attempts to determine the inverse channel characteristics using only a signal read from that channel. A Bayesian blind equalizer [3] attempts to determine the forward channel characteristics in a similar way. However, while the former technique produces an estimate of the original symbols, the Bayesian method predicts which symbol sequence best matches the signal read back. It does this by convolving all combinations of input symbols with the current estimate of the channel response and assigning them a metric of fit against the original signal. Then the first symbol of the sequence with the best metric is fed back to update the channel response estimate. This was originally developed for use in mobile telephone systems. The advantage of the Bayesian system over the inverse system is that accuracy is much higher and consequently the adaption speed is faster (by an order of magnitude). It is, however, more computationally intensive.

Neither method is really suitable for disk or tape systems because of their speed requirements. However, a credit card has a finite capacity (the entire card can be easily stored in memory) and speed is mostly unimportant.

To adapt Bayesian equalization to a card it is necessary to add velocity as a search parameter or resample the data so that the clock rate is fixed. It is also straightforward to remove any inter-track interference. Work completed so far indicates that filtering will increase the channel capacity at least twofold. Currently the processing to achieve this takes of the order 20 s on a desktop computer. Further refinement will reduce this to around 1 s in a dedicated system.

B. Feasibility

To test whether it is reasonable to estimate a channel with only a readback signal, a card written at 1500 bits/in was swiped into the system. The initial results (fig. 4 and 5) show that even with only simple processing, deficiencies in the zero crossing detector can be located. Where an error is spotted, the much slower Bayesian detection method can be employed.

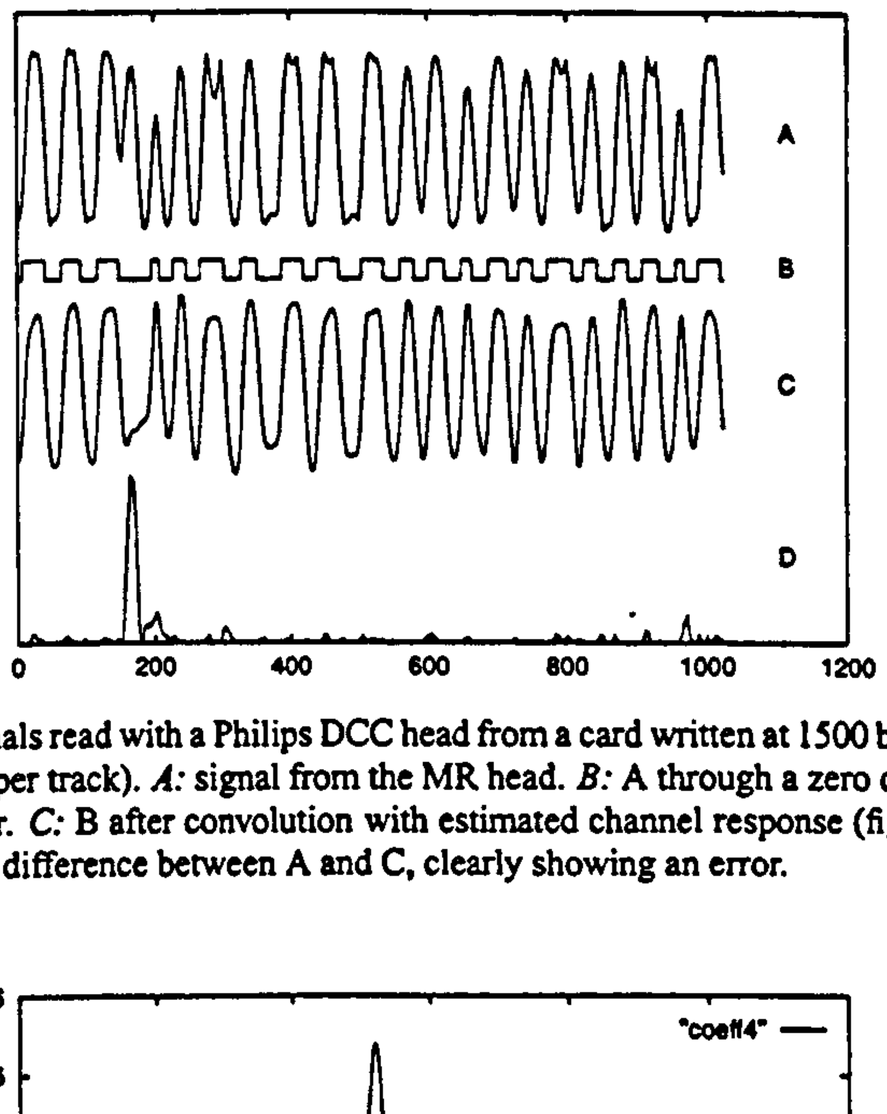


Fig. 4. Signals read with a Philips DCC head from a card written at 1500 bits/in (0.5 kByte per track). *A:* signal from the MR head. *B:* A through a zero crossing detector. *C:* B after convolution with estimated channel response (fig. 5). *D:* squared difference between A and C, clearly showing an error.

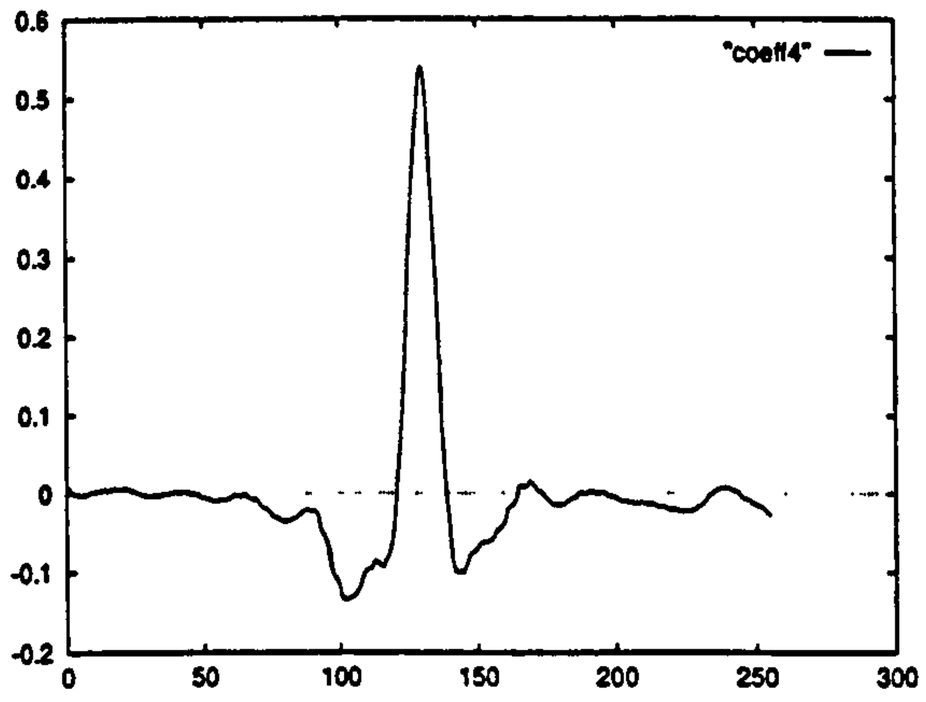


Fig. 5. Channel response, as estimated by an LMS adaptive filter from A and B in fig. 4. (No velocity compensation.)

IV. Results

With no preprocessing using an inductive read head and a software peak detector, the linear density has been increased from around 250 bits/in to 500 bits/in just by changing code (fig. 6). This translates into around 200 Bytes per track, or about 1 kByte over 4 tracks (15 kbit/in^2). No framing was used in these tests, i.e., the data started and finished abruptly at the ends of the stripe. With proper framing and a 1% raw bit error rate, 2 kBytes (30 kbit/in^2) has been stored.

The apparent bandwidth of the MR head is approximately double that of the inductive head (it has a much smaller gap). However the largest source of errors is inter-symbol interference due to the large head-medium separation: when the card is manufactured a protective coating of variable thickness is laid over the top of the magnetic layer.

Jitter in determining transition boundaries is not especially high—the card normally travels smoothly through the reader. However the head-medium separation reduces the sharpness of transitions. Fig. 7 shows an example of typical jitter in a mechanical reader.

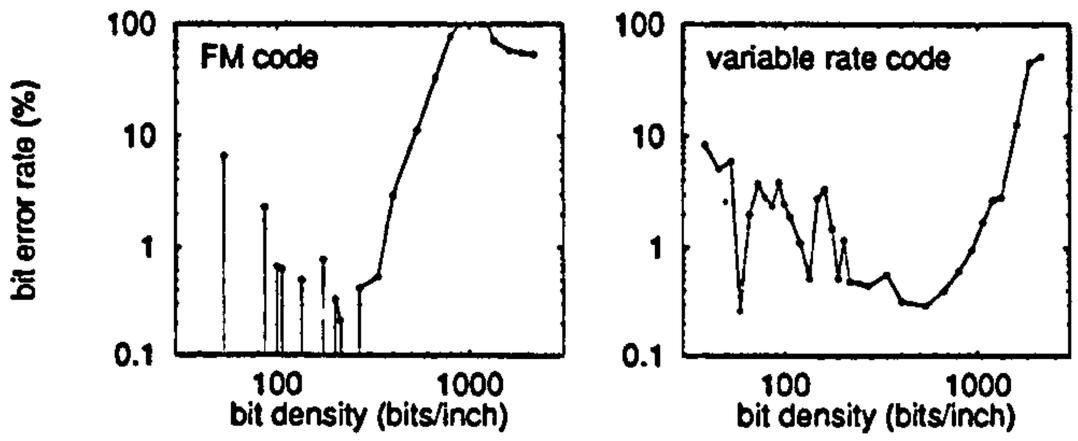


Fig. 6. Unprocessed error rates on credit cards with FM (standard) code and variable rate code. Below about 400 bits/in errors are dominated by incomplete symbols at the card edges (removed in practice by framing the data with start sentinels). Above, the dominant errors are caused by inter-symbol interference and timing noise.

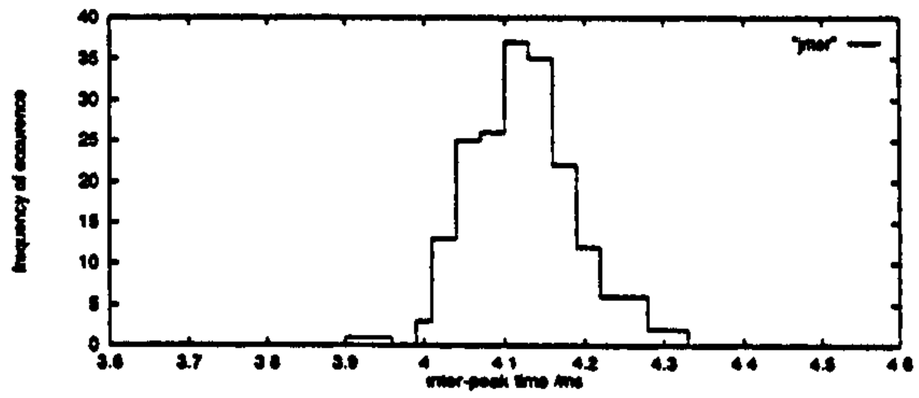


Fig. 7. Example of jitter in a mechanical card reader. Card velocity 0.12 m/s.

V. Conclusion and Discussion

The measured Shannon information capacity of a disposable credit card is of the order 80 kBytes. It is the cost, simplicity and compatibility requirements that has forced the credit card to remain firmly stuck at 140 Bytes for the last 25 years.

In changing the code used to write data to the card, an increase in capacity to 2 kBytes has been demonstrated.

By using new blind equalization methods and a multi-track MR head, it is expected that 4 kByte can be stored on a ordinary card, while still maintaining compatibility with current standard. Removing the compatibility should make it possible to store around 30 kBytes on the card (100 kbit/in^2).

Further increases in storage density will be very difficult because it is not economic or desirable to alter the card medium.

Acknowledgment

D.F.S. thanks the Royal Academy of Engineering for help with travel costs.

References

[1] "BS-7016: Recording techniques for information on identification cards," *EN 27811:1989, ISO 7811:1985*, 1989.

[2] "A picture is worth 50 words," *IEEE Spectrum*, June 1995, pp. 17.

[3] S. Chen, B. Mulgrew, S. McLaughlin, "Adaptive Bayesian equalizer with decision feedback," *IEEE Trans. Signal Processing*, September 1993, pp. 2918–2927.

Fixed Sample Rate Bayesian Detector in a Variable Speed Magnetic Channel

Daniel F. Smith and T. Donnelly and D. J. Mapps
Centre for Research in Information Storage Technology, School of Electronic, Communication and Electrical Engineering,
University of Plymouth, Drake Circus, Plymouth, PL4 8AA, UK

***Abstract*—A new design of blind equalizer is proposed that can decode a magnetic binary channel whose bit-rate is unknown. It is shown that the velocity can be estimated from information already known from a fixed delay tree search (FDTS) decision feedback equalizer (DFE) structure. Converting this detector to a Bayesian scheme with an appreciation of non-linear distortion in the read head modestly improves the velocity stability. The acquisition time for the channel speed is linear and rapid: a 50% change in velocity can be accommodated in fewer than 150 magnetic flux transitions. Results with real signals from a high-density magnetic credit card are provided.**

I. Introduction

In this application it is necessary to retrieve recorded data from a magnetic channel which has an unknown transfer function and is moving at an unknown velocity. In addition, both factors can change substantially during a short read (a few thousand symbols). Furthermore the read back signal is sampled at a fixed rate from a magneto-resistive (MR) head which has a non-linear response.

The Bayesian decision feedback equalizer (DFE) [1],[2] is well suited to fixed speed, non-stationary channels. If the DFE contains an adaptive filter then it will be slow to respond to large (a few hundred percent) changes in velocity. This paper gives a method of estimating the channel speed using the Bayesian DFE structure so that the velocity variation effects can be countered.

Additionally, the assumption that distortion is caused only by Gaussian noise is revised, and is replaced by a simple model of non-linear distortion and noise.

Finally the algorithm is applied to a signal read from the swipe of a high-density magnetic credit card [3] to assess its real-world performance.

A. System Overview

The system comprises a credit card, which has data recorded in its magnetic stripe. This card is swiped, by hand, past an MR head. The signal from the head is sampled and the waveform is stored on computer where it is decoded "off-line."

Manuscript received January 27, 1997. Revised April 24, 1997.
D. F. Smith, e-mail dfsmith@crist.plym.ac.uk, www http://crist.plym.ac.uk, T. Donnelly phone +44 1752 232516, fax +44 1752 232583.
This work is funded by the University of Plymouth. The Royal Academy of Engineering provided assistance with travel costs.

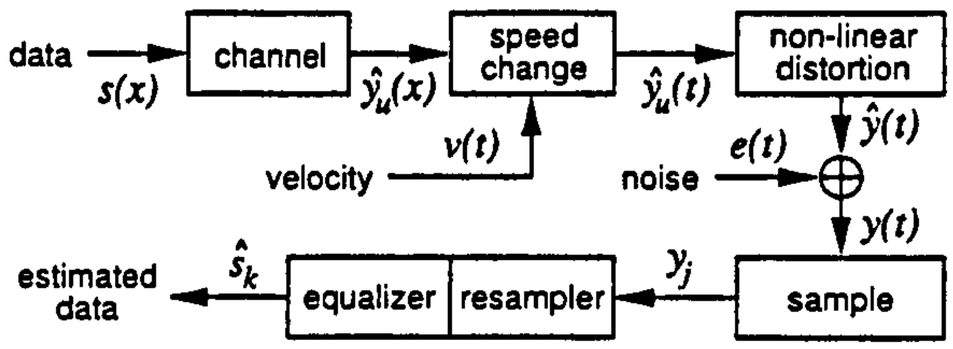


Fig. 1. Channel system: the original signal is modulated by the channel, velocity change, the read head and then system noise before being sampled and decoded.

The largest source of errors in this system is due to inter-symbol interference (ISI) principally caused by a protective layer over the magnetic medium. This produces a relatively large head to medium separation that fluctuates over the length of the card so conventional fixed equalization is not possible.

In addition, the field strength of the recorded data is often sufficient to drive the MR element into its non-linear region. This leads to small amount of peak-inversion (observed at about 10% of the signal level) for magnetizations in one direction and peak-flattening in the other.

B. Terminology

There are, necessarily, many terms and stages in this system. Fig. 1 shows the basic assumptions, the main supposition being that a change in velocity can be compensated by changing the sampling rate (i.e., that dynamic effects are minimal).

Starting at the top left of the figure, the original data, $s(x)$, are stored as flux regions on the magnetic stripe at position, x. These are ideal binary magnetizations with a constant code clock length. This data stream is modified by the channel, which is assumed to be similar to a finite impulse response (FIR) filter, to give a readback signal that is continuous in position: $s(x) \rightarrow \hat{y}_u(x)$.

When the signal is read back, at variable velocity $v(t)$, it is assumed to undergo a simple stretch $\hat{y}_u(x) \rightarrow \hat{y}_u(t)$ where the relationship between x and t is the natural $x = \int v(t)\,dt$. The signal is then distorted by the read head: $\hat{y}_u(t) \rightarrow \hat{y}(t)$. The distortion is caused by saturation of the MR head: in small fields $\hat{y}_u(t)$ is linear to $\hat{y}(t)$, but this relationship deteriorates significantly in larger fields. The distortion bears a reasonable resemblance to $\hat{y}(t) = \left(\tan^{-1}(\hat{y}_u(t) + 2.0)\right)^2$ although the exact form can only be measured from the head used.

Noise, $e(t)$, is inevitably added before the signal is sampled at a fixed rate ($\hat{y}(t) \rightarrow y(t) \rightarrow y_j$). Finally an estimate, $\hat{s}_k$, of the original data is extracted.

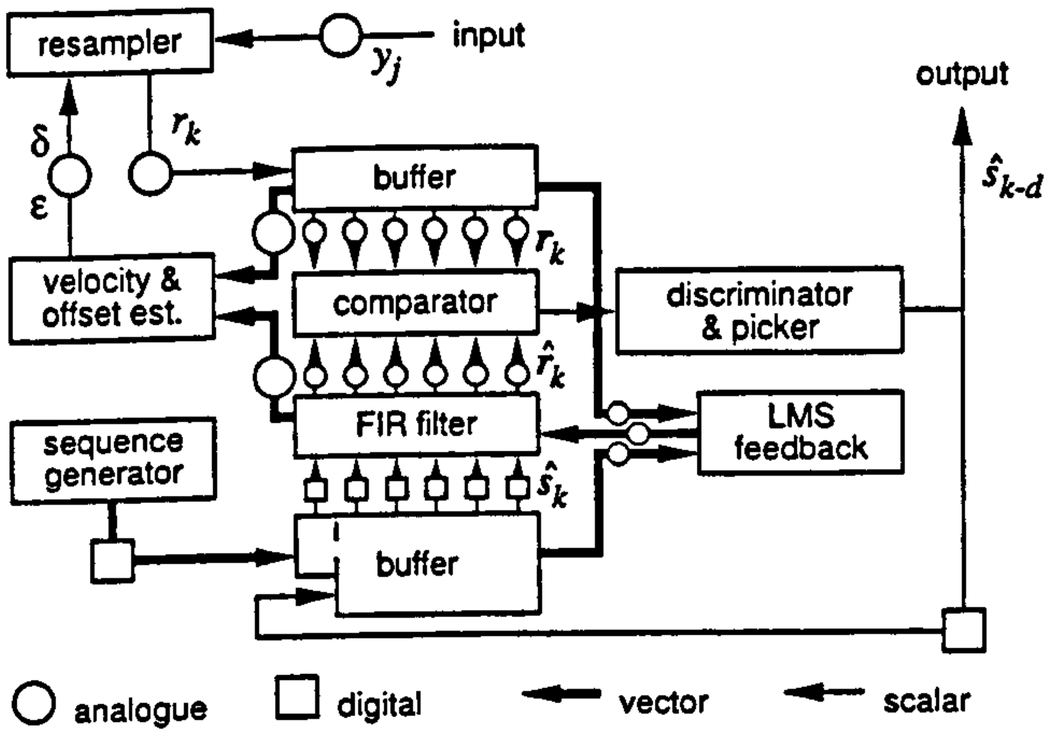


Fig. 2. Data flow in the equalizer.

II. Equalizer Structure

The equalizer follows the basic DFE design but with a resampling facility. The algorithm for determining the resampling parameters will be shown in section III. Note that limits on summations and constant factors in comparable probabilities have been omitted in this section.

A. General Structure

Referring to Fig. 2, the idea of the equalizer is to generate an analogue vector, $\hat{r}_k$, very similar to the signal recovered from the channel, r_k. Conventional inverse filters try to modify the recovered signal to give the original data, but this method modifies data to best match the signal; as follows.

A buffer, $\hat{s}_k$, contains a pattern similar to s_k as recorded in the channel. This estimation vector comprises delayed decisions from the discriminator, $\hat{s}_{k-d}, \hat{s}_{k-d-1}, \ldots$ and also a contribution from a sequence generator, which iterates over all possible $\hat{s}_k \ldots \hat{s}_{k-d+1}$ in every cycle. This gives several vectors, $\hat{s}_k^{(i)}$, to test against the (unknown) original signal, s_k.

The contents of the buffer are convolved, through an FIR filter that contains the estimated channel impulse response, to form $\hat{r}_k^{(i)}$. Each $\hat{r}_k^{(i)}$ is compared with the (known) readback signal, r_k, and the results are used to decide the new value of $\hat{s}_{k-d+1}$ before incrementing k.

Since we have a vector of channel samples and we know the probable symbols that caused the samples, the channel filter estimate can be updated very simply using the least mean squares (LMS) algorithm.

B. FDTS vs. Bayesian DFE

There are many possible forms for the comparator. The best known is probably the maximum likelihood Viterbi algorithm (MLVA), which attempts to minimize the mean square error over all time:

$$J(\hat{r}^{(i)}, r) = \sum_{\ell} \left(\hat{r}_\ell^{(i)} - r_\ell\right)^2. \tag{1}$$

Fixed delay tree search (FDTS) procedures [4] are a more practical subset of the MLVA: they search $J(\cdot)$ over a limited ℓ to form a localized log-likelihood metric against both possible values of $\hat{s}_{k-d+1}$.

A Bayesian comparator finds the sequence that was most likely to have caused the signal received. If the channel impulse response estimate is accurate and the system is synchronized then the only signal errors will be due to noise. It is common to assume that the noise, $e(t)$, is Gaussian, in which case the most likely sequence is given by

$$\eta_G = \max_i \exp\left(-\frac{J(\hat{r}^{(i)}, r)}{2\sigma^2}\right) \tag{2}$$

where σ^2 is the variance of $e(t)$. For a single comparison of $\hat{r}_k^{(i)}$, this is exactly equivalent to minimizing J, i.e., the Bayesian DFE and FDTS/MLVA are equivalent.

A minor difference between the Bayesian scheme and the MLVA is that the metric represents a (scaled) probability. Bayes' theorem[1] states that the likelihoods can be summed over all possible paths through the trellis [5]. The current decision is then based on the difference between the *probability sums* of the two top sub-paths.

However, the Bayesian DFE is a superset of the FDTS, and because because its metrics are (linear to) probabilities, there are some beneficial manipulations that can be performed with respect to non-linearities, explained below.

C. Non-linear Bayesian DFE

The FIR filter output is linear whereas the output from a typical MR head is not. This means that noise is not the only difference between $\hat{r}_k$ and r_k. In some instances it may be possible to exactly compensate for the head, but in a poorly controlled environment it is reasonable simply to manipulate the signal likelihoods.

A general comparator function can be defined,

$$\eta_f = \max_i f\left(\hat{r}^{(i)}, r\right), \tag{3}$$

to allow for any particular channel. The following simple comparator with a truncated-Gaussian error gives improved filtering by making the head response more symmetric.

$$f(\hat{r}^{(i)}, r) = \exp\left(-\sum_{\ell} g(\hat{r}_\ell^{(i)}, r_\ell)\right) \tag{4}$$

$$g(a,b) = \begin{cases} 0 & \text{if } a > +r_t \text{ and } b > +r_u \\ 0 & \text{if } a < -r_t \text{ and } b < -r_u \\ (a-b)^2 & \text{otherwise} \end{cases} \tag{5}$$

In effect this says that if the channel estimate and the channel both agree that the signal is large, then it is 'good enough' even though the actual signal levels may be disparate. This covers the saturation in the MR head, but will not cover extreme magnetic fields stronger than the MR bias field.

[1] $\text{pr}(A|B)\,\text{pr}(B) = \text{pr}(B|A)\,\text{pr}(A) \implies \text{pr}(\hat{s}_{k-d} = s_{k-d}) = \text{pr}(\hat{s}_{k-d} = s_{k-d} \text{ and } \hat{s}_k^{(1)} = s_k) + \text{pr}(\hat{s}_{k-d} = s_{k-d} \text{ and } \hat{s}_k^{(2)} = s_k) + \cdots$

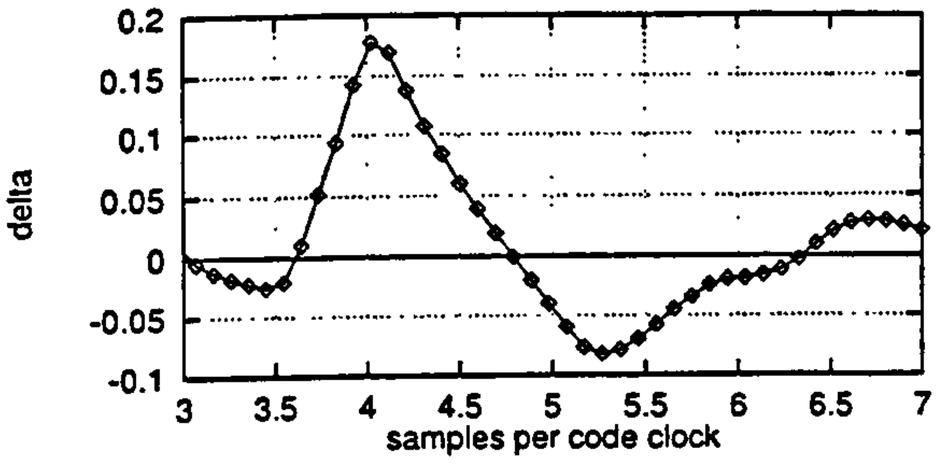


Fig. 3. Variation of δ (velocity offset) against artificially induced velocity variation. The true velocity is around 4.8 samples per clock. At velocities lower than this ($>$ 4.8 samples/clock) δ is mostly negative and vice-versa. Note that ϵ has been set to 0.

III. Speed Estimation

Suppose we have an estimate of card velocity, $\hat{v} = (1+\delta)v$ where v is the true velocity and δ is small. Also suppose that we know there might be a slight timing offset (in actual fact, a position offset) $\hat{x} = x + \epsilon$ where x is the true position and ϵ is small. From Taylor's theorem we derive the following for the sampled signal, $y(x)$.

$$\int_0^\tau \frac{dy}{dt}(vt)\left(y((1+\delta)vt+\epsilon) - y(vt)\right)dt \approx \epsilon \cdot \frac{1}{v}\int_0^\tau \left(\frac{dy}{dt}(vt)\right)^2 dt + \delta \int_0^\tau t\left(\frac{dy}{dt}(vt)\right)^2 dt \quad (6)$$

Despite appearances, this says that $Z = \epsilon X + \delta Y$ where X, Y and Z represent the values of the integrals in (6). They are obtained approximately from r and $\hat{r}$. The (seemingly redundant) $\frac{dy}{dt}(vt)$ term is included to make the integrands on right hand side of (6) positive and thus the integrals substantial. The length of the buffers is represented by τ and $y((1+\delta)vt+\epsilon)$ is a substitute for $\hat{r}_k$ and $y(vt)$ is a substitute for r_k.

Refined velocity or offset measurements can now be obtained from δ or ϵ. In practice only offset *or* speed corrections are necessary: in a constant velocity system ϵ can be used to correct phase errors whereas in this system it was sufficient to set the new value of $\hat{v} \leftarrow (1-\delta)\hat{v}$ where $\delta = Z/Y$.

Fig. 3 shows how δ varies with the difference in velocity. For small discrepancies, the value of δ is linear in mismatch, although for large differences this breaks down. Nevertheless δ still remains 'mostly positive' or 'mostly negative' depending on the direction of error.

IV. Results

There are a number of striking features in this feedback filter system. Most prominent is the way (6) locks into the velocity in a linear fashion: Fig. 4 shows the estimated speed as a function of time for a range of starting values with a $(d,k) = (1,3)$ code. The exact evolution is data dependent, with superior results for random sequences. The feedback is also exceptionally fast—for starting values within 50% of the 'correct' value, the velocity is normally locked in less than 350 code clocks (fewer than 150 magnetic transitions).

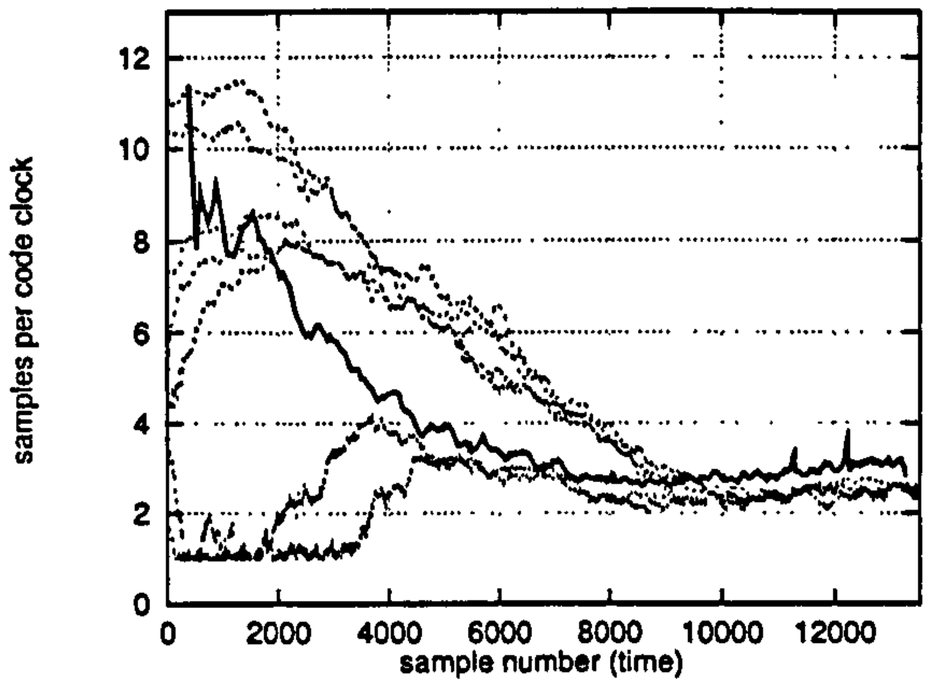


Fig. 4. Velocity acquisition from different starting estimates. The channel was sampled at between 3 and 8 samples per code clock using a $(d,k) = (1,3)$ code on a high-density credit card. Note that the algorithm blocked velocity estimates of less than 1 sample per clock. The solid line shows the card velocity read from a separate, dedicated, clock track.

The observed difference between the truncated-Gaussian error and Gaussian error metrics is subtle but worthwhile in this blind channel. The truncated error function loses lock on the signal less frequently than the pure error form.

The LMS filter is vital to the functioning of the system, but it can also cause an instability. It is possible to obtain a semi-stable velocity that is too high: the channel impulse response estimate widens from the LMS feedback so that the DFE can accurately match the expanded signal by simply modulating the duty cycle of $\hat{s}$. When the velocity estimate is too low, lock can be delayed because the FIR becomes multi-peaked.

V. Conclusion

An adaptive filter and DFE were placed in a fixed sampling rate system to equalize a variable speed channel. Using only the registers in the DFE the velocity changes in the underlying channel were removed by digital resampling. Although the system can fail, none of these failures are unreasonable given their unnatural conditions. A simple truncation adjustment to the Bayesian discriminator provided a small improvement over a purely linear system.

References

[1] S. Chen, B. Mulgrew, and S. McLaughlin, "Adaptive Bayesian equalizer with decision feedback," *IEEE Transactions on Signal Processing*, vol. 41, pp. 2918–2927, September 1993.

[2] G.-k. Lee, S. B. Gelfand, and M. P. Fitz, "Bayesian decision feedback techniques for deconvolution," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 155–166, January 1995.

[3] D. Smith, T. Donnelly, and D. Mapps, "High density storage on a magnetic stripe card," *IEEE Transactions on Magnetics*, vol. 32, pp. 4025–4027, September 1996.

[4] J. Moon and L. R. Carley, "Performance comparison of detection methods in magnetic recording," *IEEE Transactions on Magnetics*, vol. 26, pp. 3155–3172, November 1990.

[5] K. Abend and B. D. Fritchman, "Statistical detection for communication channels with intersymbol interference," *Proceedings of the IEEE*, vol. 58, pp. 779–788, May 1970.

Appendix B

Code properties and the FSTM

The finite state transition matrix (FSTM) is a compact way of defining a code that is also useful for calculating some properties of that code, e.g., the code efficiency and power frequency spectrum. This appendix derives the FSTM and some standard properties, then tersely follows the calculation of the frequency spectra of some generic (d, k) codes.

B.1 Code efficiencies

To derive the efficiency of a code, the code must be described mathematically. From this first the capacity and then the efficiency of the code can be calculated.

B.1.1 The State Machine

Although magnetic domains on magnetic media are approximately discrete and stochastic in form, containing a three-dimensional fixed magnitude magnetic vector, technical limitations force head-medium systems to treat the medium as a one-dimensional two-state memory. Furthermore, it is helpful to limit the complexity by quantizing the dimension, and treat the memory as simply a stream of 0 or 1 bits, corresponding (nominally) to magnetization forwards or backwards in a group of domains.

A coding device can be considered to be an algorithm that takes a stream

of input bits and produces a stream of output bits. The output bits will be of a form suitable to lay down and read back from the magnetic medium. For instance, the medium may put restraints on the maximum transition density in the bit stream, timing recovery may put constraints on the minimum transition density and threshold determination may require the code to have a roughly equal number of 1s as 0s.

One desirable property of a coding algorithm is efficiency: the number of output bits should be as small as possible while still uniquely coding the input bits. Calculating the efficiency is not a trivial matter for most codes.

The coding algorithm can be defined in terms of a (finite) state machine. (See figure B.1 for example.) The connectivity of the m-state machine can be completely represented by a $m \times m$ matrix, known as the finite state transition matrix. If the FSTM is A then A_{ij} represents the number of routes from state j to state i. Each state change takes 1 unit of time. (In the literature the matrix is usually written in transpose form, A^T).

m = no. states

A = FSTM

B.1.2 Calculation of code capacity

The capacity of a code is calculated from the expression for the number of different possible routes around its state machine, N, in a given number of steps (n). The number of bits that can be encoded in n steps is thus $\log_2 N(n)$.

n = no. steps
N(n) = no. paths

The capacity, C, is taken as the ratio of coded bits to state steps: i.e.,

$$C = \lim_{n \to \infty} \frac{\log_2 N(n)}{n}. \tag{B.1}$$

C = capacity

The capacity represents the relative efficiency of the code compared with NRZ code (a straight binary bit stream) in terms of output bits.

B.1.3 Calculation of $N(n)$

Consider a machine in state s. This can be expressed as a $1 \times m$ column vector, v, where $s_i = 1$ for $i = s$ and $s_i = 0$ otherwise. For instance, the following initial

v = initial states

state vector has $s = 1$ (the vector is indexed from zero).

$$\boldsymbol{v} = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

In fact, $\boldsymbol{v}$ can represent a collection of state machines, with v_0 machines in state 0, v_1 machines in state 1, etc..

Given a machine in state $\boldsymbol{v}$, the machine's state after 1 step is given by

$$\boldsymbol{w} = \mathrm{A}.\boldsymbol{v} \tag{B.2}$$

w = *new state*

Both $\boldsymbol{v}$ and $\boldsymbol{w}$ are known as occupancy vectors. If the machine's state can bifurcate, then the new state ($\boldsymbol{w}$) reflects this. In the above example, if the machine can go from state 1 to state 0 or 2, then $\boldsymbol{w}$ would be,

$$\boldsymbol{w} = \begin{pmatrix} . & 1 & . & \dots \\ . & 0 & . & \dots \\ . & 1 & . & \dots \\ . & 0 & . & \dots \\ \vdots & \vdots & & \ddots \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ \vdots \end{pmatrix}$$

This procedure also works with collections of state machines.

u = *unitary vector*

Now the calculation of $N(n)$ is straightforward.

$$N(n) = \sum_{i=0}^{m-1} w_i(n) = \boldsymbol{u}^T \boldsymbol{w}(n) \tag{B.3}$$

where $\boldsymbol{u}^T = \begin{pmatrix} 1 & 1 & 1 & \dots \end{pmatrix}$ is the unitary vector of 1s and

$$\boldsymbol{w}(n) = \mathrm{A}^n \boldsymbol{v} \tag{B.4}$$

In the limit $n \to \infty$ the initial state of v is unimportant. (Consider the minimal consequences for $w(n)$ if $v = w(100)$ initially.)

To calculate $\mathbf{A}^n$, the eigenvalues, $\lambda_{1\ldots m}$, and eigenvectors, $e_{1\ldots m}$, of $\mathbf{A}$ are needed. The $m \times m$ eigenmatrix

λ = *eigenvalues of* $\mathbf{A}$
e = *eigenvectors of* $\mathbf{A}$
Λ = *eigenmatrix of* $\mathbf{A}$

$$\Lambda = \begin{pmatrix} e_1 & e_2 & \cdots & e_m \end{pmatrix} \tag{B.5}$$

is constructed from the eigenvectors. Now, from the properties of these vectors the following can be written

$$\mathbf{A}.\Lambda = \mathbf{A} \cdot \begin{pmatrix} e_1 & e_2 & \cdots \end{pmatrix} = \begin{pmatrix} \lambda_1 e_1 & \lambda_2 e_2 & \cdots \end{pmatrix} \tag{B.6}$$

hence

$$\mathbf{A}^n.\Lambda = \begin{pmatrix} \lambda_1^n e_1 & \lambda_2^n e_2 & \cdots \end{pmatrix} \tag{B.7}$$

and

$$\mathbf{A}^n = \begin{pmatrix} \lambda_1^n e_1 & \lambda_2^n e_2 & \cdots \end{pmatrix} \cdot \Lambda^{-1} \tag{B.8}$$

where Λ^{-1} is the inverse* of matrix Λ.

Now the form of $\mathbf{A}^n$ is a polynomial matrix in λ_i. Consequently, $w(n)$ is a polynomial vector in λ_i (B.4) and at last we can see from (B.3) that

$$N(n) = \sum_{j=1}^{m} c_j \lambda_j^n \tag{B.9}$$

where the c_j are the polynomial coefficients calculated above and summed. As $n \to \infty$ the form of $N(n)$ is dominated by the largest (positive) value of λ, (nominally λ_1).

λ_1 = *largest* λ

$$\lim_{n\to\infty} N(n) \to c_1 \lambda_1^n \tag{B.10}$$

*Occasionally the inverse of Λ does not exist. However the following argument still works because Λ^{-1} is constant (not a function of n). Consider dividing (B.7) by λ_1^n and then taking the limit $n \to \infty$.

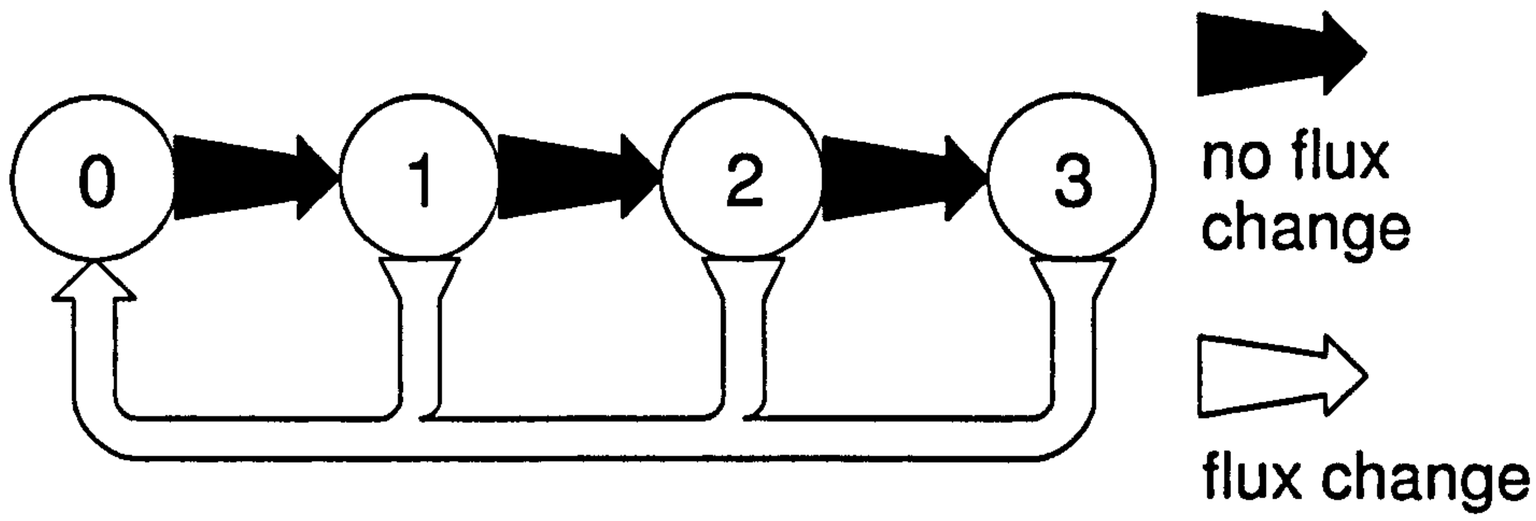


Figure B.1: State machine for $(d, k) = (1, 3)$.

Putting this back in (B.1) we find

$$\begin{aligned} C &= \lim_{n\to\infty} \frac{1}{n} \log_2(c_1 \lambda_1^n) \\ &= \lim_{n\to\infty} \frac{1}{n} (n \log_2 \lambda_1 + \log_2 c_1) \\ &= \log_2 \lambda_1 \end{aligned} \tag{B.11}$$

B.1.4 Example capacity

A $(d, k) = (1, 3)$ code (figure B.1) has the following FSTM.

$$\mathbf{A} = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

The characteristic equation of $\mathbf{A}$ is $\lambda^4 - \lambda^2 - \lambda - 1 = 0$. The largest root is $\lambda_1 = 1.465$, hence the capacity of a $(d, k) = (1, 3)$ code is $\log_2 1.465 = 0.551$.

A $(d, k) = (0, 1)$ code (figure B.2) has the following FSTM.

$$\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$$

The characteristic equation of $\mathbf{A}$ is $\lambda^2 - \lambda - 1 = 0$. The largest root is $\lambda_1 = 1.618$,

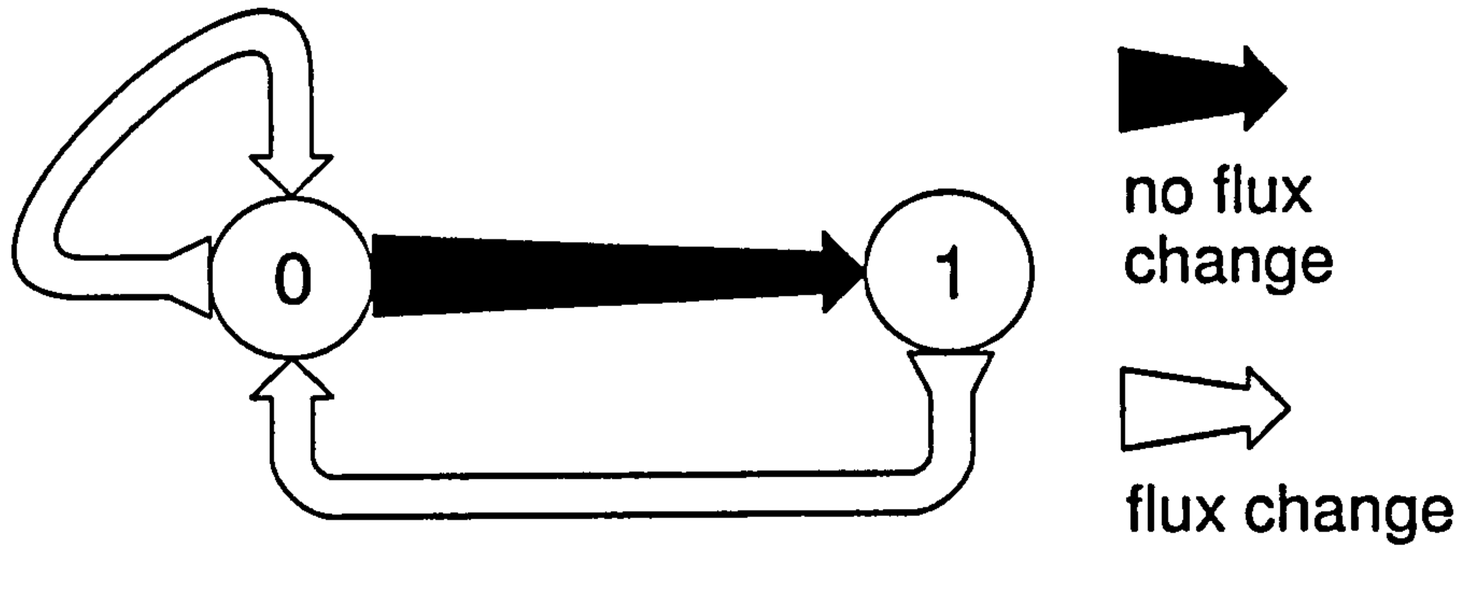


Figure B.2: State machine for $(d, k) = (0, 1)$.

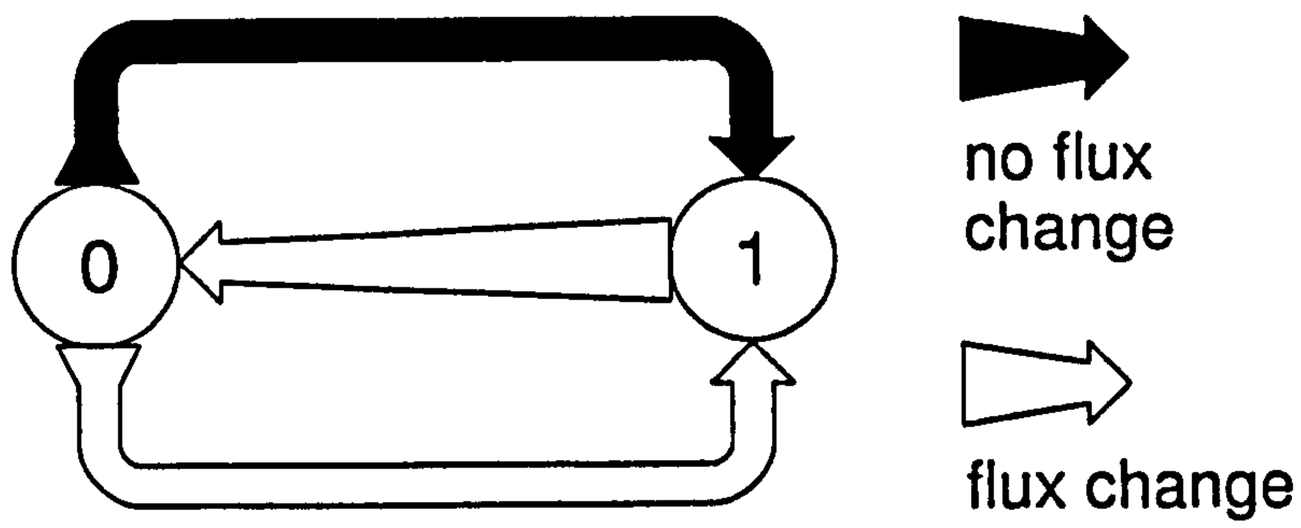


Figure B.3: State machine for FM code.

hence the capacity of a $(d, k) = (0, 1)$ code is $\log_2 1.618 = 0.694$.

FM code is a $(d, k) = (0, 1)$ code that has been used extensively (figure B.3).

$$\mathbf{A} = \begin{pmatrix} 0 & 1 \\ 2 & 0 \end{pmatrix}$$

The characteristic equation of $\mathbf{A}$ is $\lambda^2 = 2$. The largest root is $\lambda_1 = 1.414$, hence the capacity of FM code is $\log_2 \sqrt{2} = 0.5$.

B.1.5 Capacity of code classes

Codes are often classified as run-length limited (RLL) (d, k), $(d, k; m, n; r)$ or charge-constrained RLL (CCRLL) $(d, k, c; m, n; r)$ [73]. The letters refer to code constraints. In the following list, a 1 refers to a magnetic flux change, and a 0

no flux change in the data stream.

d the minimum number of 0s between 1s.

k the maximum number of 0s between 1s.

RDS: running digital sum

c the maximum value of RDS: i.e., the magnetic charge constraint.

m the input stream symbol size.

n the output stream symbol size (i.e., for m input bits n output bits are generated).

r the number of code tables that are used to translate m to n.

Tang [89] showed that the characteristic equation of a code that was only limited in its (d, k) restraint could be written in closed form.

$$\lambda^{k+2} - \lambda^{k+1} - \lambda^{k-d+1} + 1 = 0 \tag{B.12}$$

Norris [73] and Chien [17] both derive and tabulate some properties of selected charge constrained codes.

B.1.6 Code efficiency

The efficiency of a code is normally expressed as the ratio of the proper code capacity to that of a general code with the same (d, k) restraint.

$$E = \frac{C_{\text{code}}}{C_{(d,k)}} \tag{B.13}$$

Hence, the efficiency of the FM code described above is $\frac{0.5}{0.69} = 72\%$.

B.1.7 Variable rate code

The state machine for variable rate code ($s_0 = 2$, $S = 0, 1, 2$, $(d, k) = (1, 3)$, described in section 3.3, page 67) is shown in figure B.4 and has code properties as shown in figure B.5.

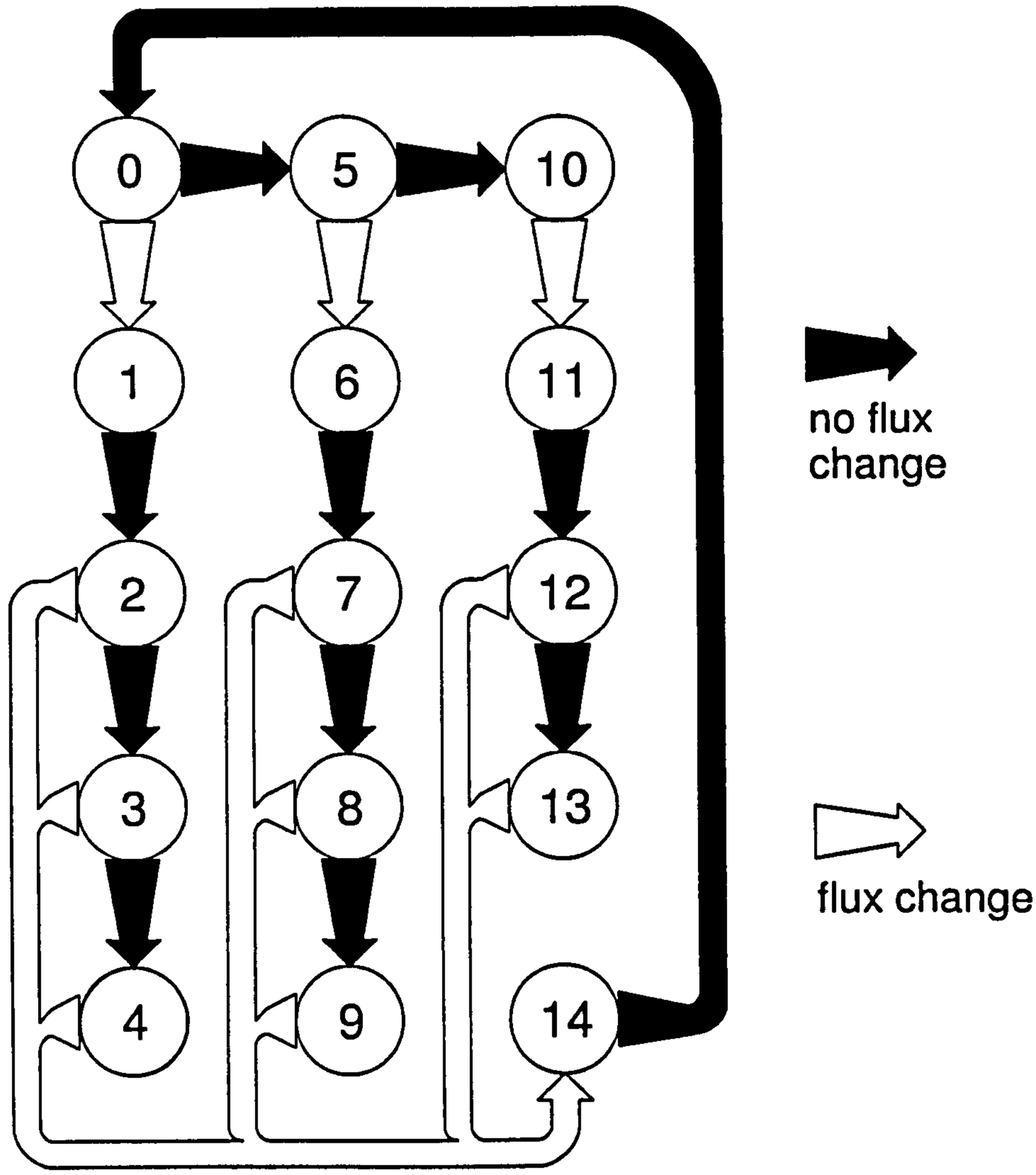


Figure B.4: State machine for variable rate code.

The efficiency of this code is thus $\frac{0.539}{0.551} = 97.7\%$. Other popular $(d, k) = (1, 3)$ codes generally have efficiencies of around 91%, making variable rate code almost 7% more efficient than other codes with the same (d, k) properties.

It should be noted that it is easy to make a 100% efficient variable rate code that is also useful. E.g., $(d, k) = (1, 4)$, $s_0 = 2$, $S = 0, 1, 2, 3$ or $(d, k) = (0, 1)$, $s_0 = 1$, $S = 0, 1$.

$$\begin{pmatrix}
. & . & . & . & . & . & . & . & . & . & . & . & . & . & 1 \\
1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & 1 & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & 1 & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & 1 & . & . & . & . & . & . & . & . & . & . & . \\
1 & . & . & . & . & . & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & 1 & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & 1 & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & 1 & . & . & . & . & . & . \\
. & . & . & . & . & 1 & . & . & . & . & . & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & 1 & . & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & 1 & . & . & . \\
. & . & . & . & . & . & . & . & . & . & . & . & 1 & . & . \\
. & . & 1 & 1 & 1 & . & . & 1 & 1 & 1 & . & . & 1 & 1 & .
\end{pmatrix}$$

Characteristic equation: $\lambda^8 (\lambda^7 - \lambda^3 - 2\lambda^2 - 3\lambda - 2) = 1$
Largest eigenvalue: $\lambda_1 = 1.452, \quad \log_2 \lambda_1 = 0.539.$

Figure B.5: Finite state transition matrix for variable rate code.

B.2 Code power spectra

It is now possible to find the code's power spectrum by using its FSTM to derive its autocorrelation spectrum.

B.2.1 Autocorrelation and the power spectrum

Take a code sequence as written by a recording head: $XXXYYXYXYXYX$ where X indicates magnetization in one direction and Y magnetization in the other. The (d, k) constraint says the X and Y symbols must be in groups of $d+1, d+2, \ldots, k+1$. For convenience, a run of similar symbols will be called a symbol group.

D = delay operator
$A(D) = A_1 D + A_2 D^2 + \cdots$

A whole sequence can be represented in one function, A, by writing

$$A(D) = \sum_{i=-\infty}^{\infty} A_i D^i \tag{B.14}$$

where D represents a delay of one time unit* and $A_i = -1$ means the ith symbol is an X and $A_i = +1$ means the ith symbol is a Y.

The spectrum of $A(D)$ as impulses can be derived from its autocorrelation (the Weiner Khintchine theorem) following the conventions in [33, 104].

$S(Y, D)$ = *spectrum*
$R_j(Y)$ = *autocorrel.*

$$S(A, D) = \sum_{j=-\infty}^{\infty} R_j(A) D^j \tag{B.15}$$

$R_j(A)$ is the jth discrete autocorrelation coefficient:

$$R_j(A) = \frac{\sum_i A_i A_{i+j}}{\sum_i 1}. \tag{B.16}$$

The sums in the numerator and denominator are over the same limits and the normalization is to convert the units of R to expectation values,

$$R_j(A) = \sum_{i=\text{all possible } A_x\, A_y} i \,\mathrm{pr}(A_0 A_j = i) \tag{B.17}$$

The pr operator is defined slightly strangely, with a 'sliding origin' so that $\mathrm{pr}(K = k)$ reads as the probability that *any* K_i is k:

$$\mathrm{pr}(A_0 A_j) = \lim_{L\to\infty} \frac{1}{2L+1} \sum_{i=-L}^{L} (A_{i+0} A_{i+j} = i) \tag{B.18}$$

(here $(x = y)$ has the value 1 if $x = y$ and value 0 is $x \neq y$).

In order to recover the impulse frequency spectrum, D is replaced with $e^{i2\pi fT}$ where (in this context) $i^2 = -1$, f is frequency and T is the delay introduced by D. To convert A's spectrum from impulses to a more realistic write signal, the code sequence is convolved with a top-hat function, P_T

$i = \sqrt{-1}$
f = *frequency*
T = *D's delay*

$$P_T = \begin{cases} 1, & \text{if } 0 \leq t < T \\ 0, & \text{otherwise} \end{cases} \tag{B.19}$$

where t is (continuous) time and T is the symbol period. Hence the new spec-

*Similar to a z transform in signal processing, $D \sim z^{-1}$.

trum looks like this

$$S_{\text{cont}}(A, f) = \text{sinc}^2(\pi f T) T^2 S(A, e^{i2\pi f T}) \tag{B.20}$$

where $\text{sinc}(x) = \frac{\sin(x)}{x}$.

$X(D) = \frac{1-D}{2} A(D)$

The expression

$$X(D) = \frac{1}{2}(1 - D)A(D) \tag{B.21}$$

$$\Rightarrow \quad S(X, D) = \frac{1}{4}(2 - D - D^{-1})S(A, D) \tag{B.22}$$

will be needed later: $X(D)$ appears as a stream of alternating 1s and −1s as coefficients of D, spaced by $d \ldots k$ 0s.

B.2.2 The run-length matrix, $G(D)$

The FSTM is closely related to the run-length matrix. The run-length matrix stores information on symbol changes whereas the FSTM stores information on internal state changes. The following 'states' refer to run-length matrix states, not FSTM states.

Consider the well known MFM (modified frequency modulation) code (figure B.6) which has a fixed period bit cell and a central flux transition for a 1 while a cell-start flux transition is added, if needed, to create a $(d, k) = (1, 3)$ code.

To convert from a state machine to a run-length state machine, the pathways are changed to variable length, and states are only preserved after a flux change. Hence, after the conversion, there is a single flux change at the end of each pathway. This transformation is shown for MFM in figure B.7. For a generic (d, k) code there is only one state, e.g., figure B.8.

The run-length matrix, also known as the one-step state-transition matrix, is derived from the run-length state machine as follows:

$$G_{pq}(D) = \sum_i D^i \,\text{pr}(\text{transition from state } q \text{ to state } p \text{ in } i \text{ time units}) \tag{B.23}$$

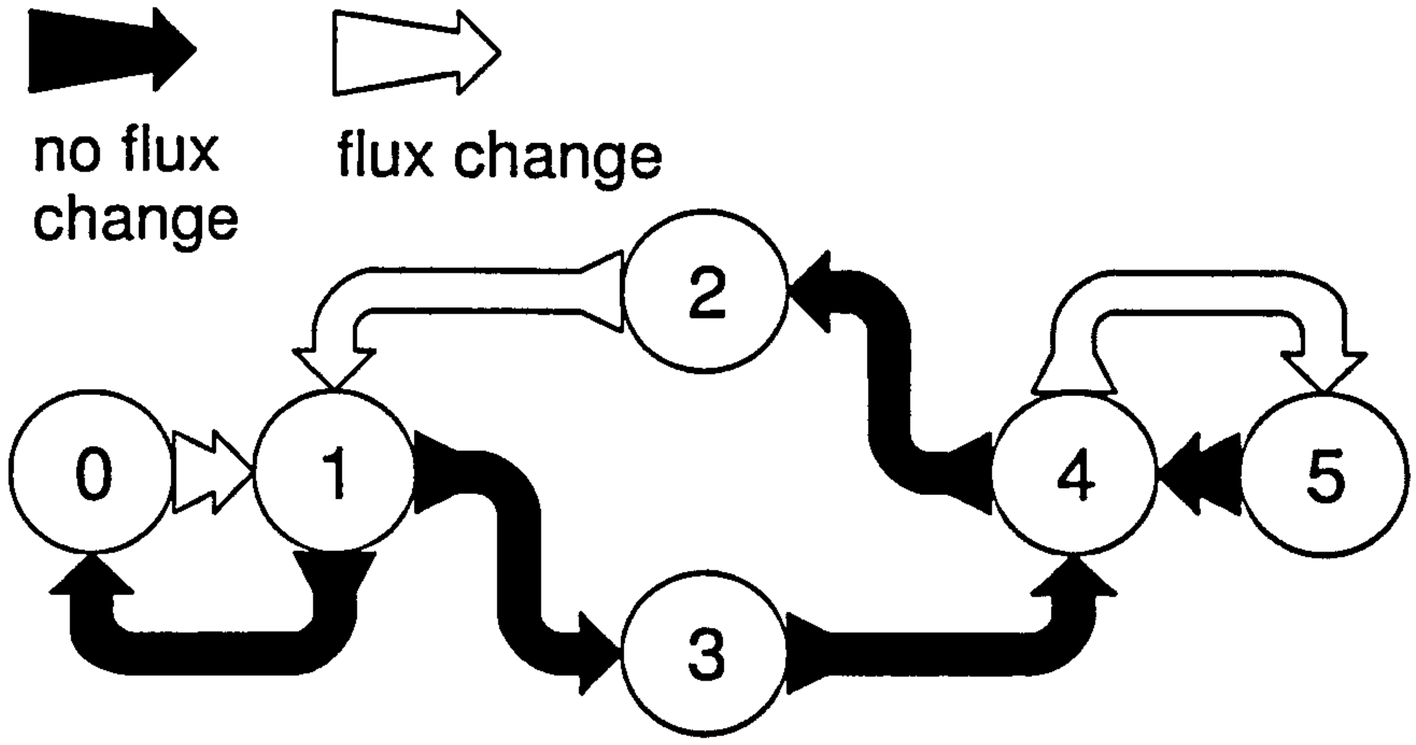


Figure B.6: State machine for MFM code.

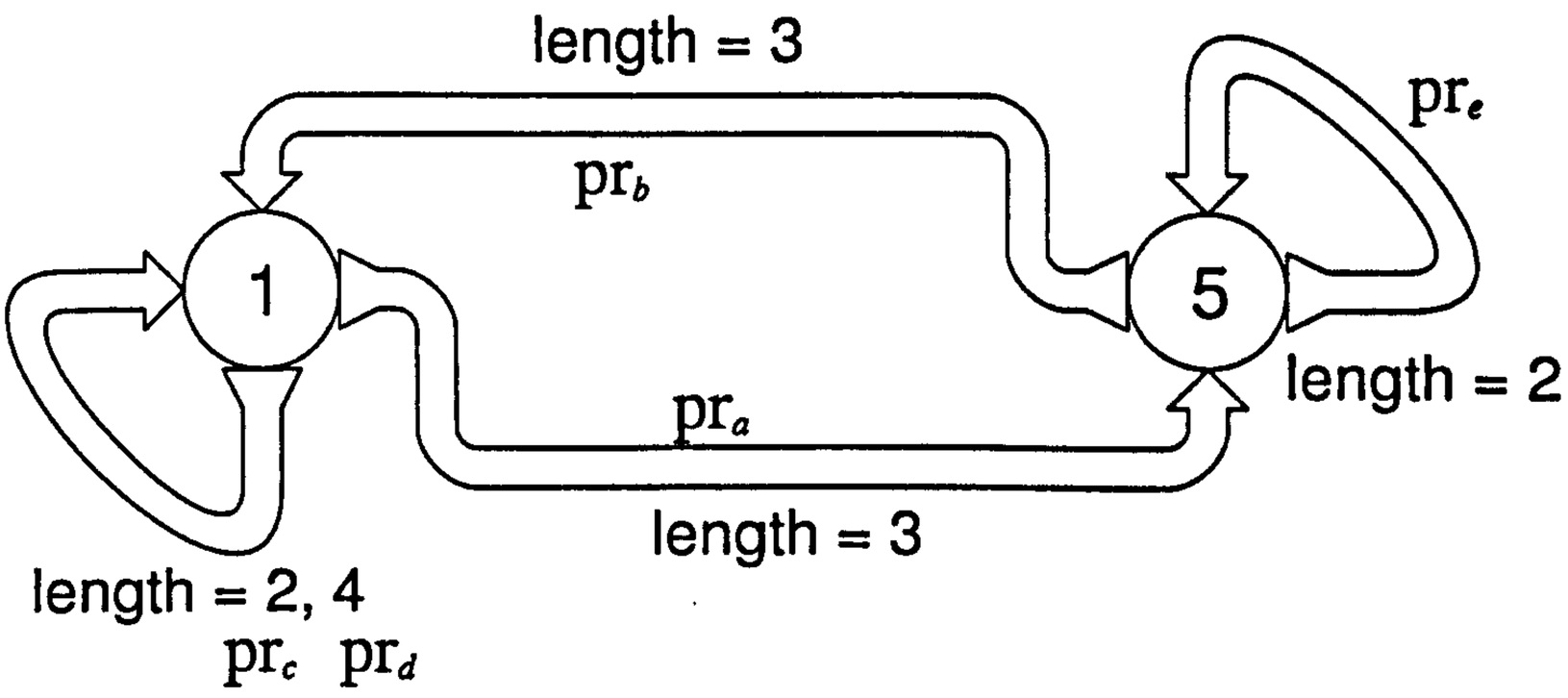


Figure B.7: Conversion of the MFM state machine into run-length state machine.

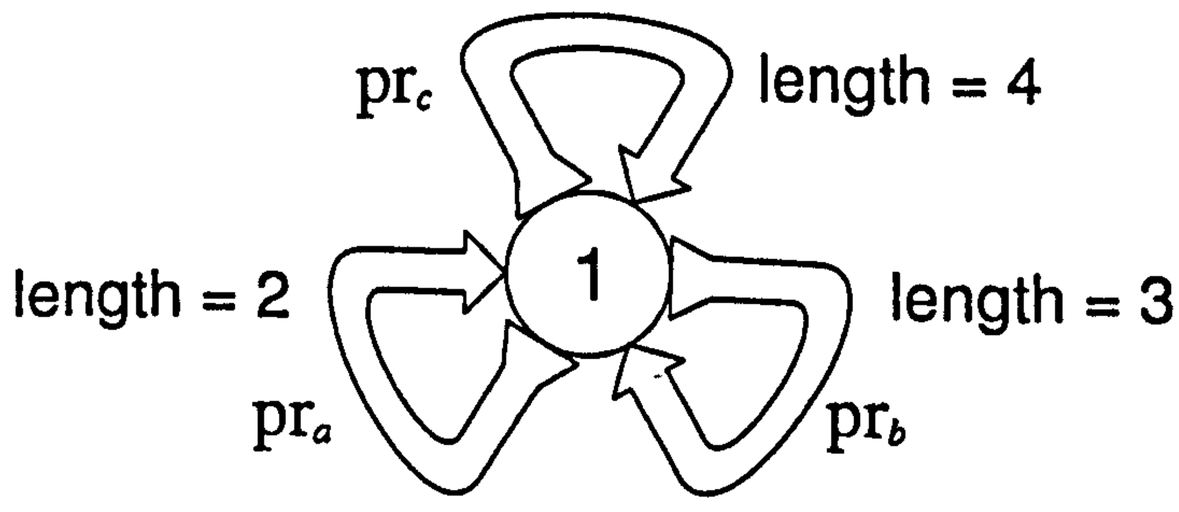


Figure B.8: Run-length state machine for $(d, k) = (1, 3)$.

with

$$\sum_i \sum_p \text{pr}(\text{transition from state } q \text{ to state } p \text{ in } i \text{ time units}) = 1 \quad \text{for all } q. \quad \text{(B.24)}$$

For example, MFM (figure B.7) with redundant states removed

$$\mathbf{G}(D) = \begin{pmatrix} \text{pr}_c D^2 + \text{pr}_d D^4 & \text{pr}_b D^3 \\ \text{pr}_a D^3 & \text{pr}_e D^2 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} D^2 + \frac{1}{4} D^4 & \frac{1}{2} D^3 \\ \frac{1}{4} D^3 & \frac{1}{2} D^2 \end{pmatrix}$$

(probabilities taken from Gallopoulos [33]).

With a $(d, k) = (1, 3)$ code with only 1 state (figure B.8)

$$G(D) = \text{pr}_a D^2 + \text{pr}_b D^3 + \text{pr}_c D^4$$

With unoptimized variable rate code and random data, $\text{pr}_a = \text{pr}_b = pr_c = \frac{1}{3}$ or generally $\text{pr}_{a\ldots} = \frac{1}{1+k-d}$. However, if the data stream is optimized, it tends to be more maxentropic, with probabilities governed by a partition function, derived below.

This argument is similar to the Z or Ξ partition function from quantum thermodynamics (e.g., [60]). Entropy must be maximized for a given code sequence length. The entropy measurement is based on the number of ways of arranging a sequence. Let the symbol W represent the number of ways of arranging N symbol groups. There are n_i symbol groups of length i so that

W = combinations
N = total symbols
n_m = no. sym. groups
L = total length

$$\sum_m n_m = N \quad \text{(B.25)}$$

and the total length of the symbols and the probability of symbol change is

$$L = \sum_m m n_m \quad \text{(B.26)}$$

$$\text{pr}(1) = \frac{N}{L} \quad \text{(B.27)}$$

From combinational theory and Sterling's approximation $\ln n! \approx n\,(\ln n - 1)$ for

large n,

$$\ln W = \ln \frac{N!}{n_1!n_2!n_3!\ldots} \approx N \ln N - \sum_m n_m \ln n_m. \tag{B.28}$$

Now the stationary points can be found

$$d(\ln W) = \sum_m (\ln N - \ln n_m)\, dn_m = 0 \tag{B.29}$$

and also the expression for the differential of the total length (which is constant)

$$dL = \sum_m m dn_m = 0. \tag{B.30}$$

From the method of undetermined multipliers it is found that

$$\ln N - \ln n_m = \mu m \quad \Rightarrow \quad n_m = e^{\ln N - \mu m} = N e^{-\mu m}. \tag{B.31}$$

And so μ can be found from (B.25)

$$N = N \sum_m e^{-\mu m} \tag{B.32}$$

which for a (d, k) code reduces to

$$e^{\mu(k+2)} - e^{\mu(k+1)} - e^{\mu(k-d+1)} + 1 = 0 \tag{B.33}$$

using the identity $\sum_0^m x^m = \frac{1-x^{m+1}}{1-x}$. This can be solved to give the final value of μ ($e^\mu \neq 1$ because it's a factor).

In the case of a $(d, k) = (1, 3)$ code, $e^\mu = 1.466$. $\mathrm{pr}_a = \frac{n_2}{N} = 0.47$, $\mathrm{pr}_b = \frac{n_3}{N} = 0.32$ and $\mathrm{pr}_c = \frac{n_4}{N} = 0.22$.

B.2.3 The $\Phi_l(D)$ function

To derive the power spectrum of a read code, it is useful to define the function

$$\Phi_l(D) = \sum_{j=1}^{\infty} D^j \,\mathrm{pr}(\text{no. symbols in } l \text{ symbol groups} = j) \tag{B.34}$$

I.e., if the system is in a probability state, $U(D)$, then

$$\Phi_l(D)U(D) = \text{new state of system after } l \text{ symbol groups.} \tag{B.35}$$

$\Phi_l(D)$ is useful because it is easily generated from the run-length matrix, as follows.

First, define a starting condition vector, v, where v_i is the probability that the

v = initial state

run-length process will start in state i. Note $\sum_i v_i = 1$.

After one symbol group, the process will be condition

$$w = \mathbf{G}(D).v \tag{B.36}$$

and after l symbol groups the process will be condition

$$w = \mathbf{G}^l(D)v. \tag{B.37}$$

It is not important which state w is finally in, so in terms of the run-length matrix

$$\Phi_l(D) = u^T\mathbf{G}^l(D)v \tag{B.38}$$

where u is once again the unitary vector.

B.2.4 The power spectrum and $\Phi_l(D)$

Recall the function $X(D)$, which might look something like

$$X(D) = \cdots + (-1)D^{-1} + 1 + 0D + 0D^2 + (-1)D^3 + 0D^4 + 1D^5 + \cdots$$

(alternating 1s and -1s separated by $d \ldots k$ 0s).

Using the properties that $R_j(Y) = R_{-j}(Y)$ and $X_iX_j \in \{-1, 0, 1\}$ with (B.15) and (B.17) the expression for the spectrum can be written

$$S(X, D) = \sum_{i=-\infty}^{-1} R_i(X)D^i + R_0(X) + \sum_{i=1}^{\infty} R_i(X)D^i \tag{B.39}$$

$$= \quad R_0(X) + \sum_{i=1}^{\infty} R_i(X)\left(D^i + D^{-i}\right). \tag{B.40}$$

The form $R_0(X) = \mathrm{pr}(X_0 \neq 0)$ shall be written as $\mathrm{pr}(1)$—the probability that a coefficient in $X(D)$ is non-zero—for brevity. Notice

$$R_i(X) = \mathrm{pr}(X_0 X_i = 1) - \mathrm{pr}(X_0 X_i = -1) \tag{B.41}$$

and also, because the 1s and −1s alternate with each symbol group

$$\mathrm{pr}(X_0 X_i = +1) = \mathrm{pr}(1)\,\mathrm{pr}(\text{even no. of symbol groups from } X_0 \text{ to } X_i) \tag{B.42}$$

$$\mathrm{pr}(X_0 X_i = -1) = \mathrm{pr}(1)\,\mathrm{pr}(\text{odd no. of symbol groups from } X_0 \text{ to } X_i) \tag{B.43}$$

for any $i \geq 1$ hence from (B.41)

$$\frac{\sum_{j=1}^{\infty} R_j(X) D^j}{\mathrm{pr}(1)} \quad = \quad \sum_{j=1}^{\infty} \mathrm{pr}(\text{even no. groups}) D^j - \mathrm{pr}(\text{odd no. groups}) D^j \tag{B.44}$$

$$= \quad -1 \quad + \sum_{l=0,2,4,\ldots} \Phi_l(D) \quad - \sum_{l=1,3,\ldots} \Phi_l(D). \tag{B.45}$$

Substituting this back into B.40 the useful equation below appears

$$S(X,D) = \mathrm{pr}(1)\left\{-1 + \sum_{l=0,2,\ldots}\left[\Phi_l(D) + \Phi_l(D^{-1})\right] - \sum_{l=1,3,\ldots}\left[\Phi_l(D) + \Phi_l(D^{-1})\right]\right\} \tag{B.46}$$

At last it is possible to express the power spectrum in terms of $G(D)$ from (B.38) and the identity $\sum_0^{\infty} \mathbf{X}^{2i} = (\mathbf{I} - \mathbf{X}^2)^{-1}$ as long as $\mathbf{X}^{2i}$ is convergent:

$$S(X,D) = \mathrm{pr}(1) \cdot u^T \left\{\left(\mathbf{I} + \mathbf{G}(D)\right)^{-1} + \left(\mathbf{I} + \mathbf{G}(D^{-1})\right)^{-1} - \mathbf{I}\right\} v \tag{B.47}$$

B.2.5 Power spectra of (d, k) codes

For a (d, k) code, $\mathbf{G}(D) = G(D)$, $u^T = (1)$ and $v = (1)$, hence

$$S(X,D) = \mathrm{pr}(1) \cdot \left\{\frac{1}{1 + G(D)} + \frac{1}{1 + G(D^{-1})} - 1\right\} \tag{B.48}$$

and

$$S(X,f) = \mathrm{pr}(1) \cdot \left\{ \frac{1}{1+G_f(f)} + \frac{1}{1+G_f(-f)} - 1 \right\} \cdot T^2 \,\mathrm{sinc}^2(\pi f T) \qquad \text{(B.49)}$$

where

$$G_f(f) = G(e^{i2\pi fT}). \qquad \text{(B.50)}$$

$\Re(x)$ = *real part of* x

Converting this back into the write signal spectrum gives (using $\Re(x)$ to denote the real part of x)

$$S(A,f) = \frac{1}{\sin^2(\pi f T)} S(X,f) \qquad \text{(B.51)}$$

$$S(A,f) = \frac{\mathrm{pr}(1)}{\pi^2 f^2} \cdot \left\{ \frac{1+|G_f(f)|^2}{1+2\Re(G_f(f))+|G_f(f)|^2} \right\}. \qquad \text{(B.52)}$$

The expressions for $G(D)$ and pr(1) of some (d,k) codes are tabulated in table B.1 and spectra are shown in figures B.9, B.10, B.11 and B.12.

It is noticeable that the $(0,x)$ codes have a rounded spectrum, well matched to a normal magnetic channel. However, at higher densities, this means that a large proportion of the high-frequency spectrum will be attenuated, giving higher distortion and hence higher error rates. The $(> 0,x)$ codes on the other hand place most of their energy into lower frequencies, giving better performance at high density. However, the codes with a wider d–k gap also have a more spread out spectrum, so the objective in matching the code to the channel is to find a code with most energy in the desired band, and also matching the shape of the channel (i.e., large spectral peaks are probably not desirable).

		uniform		
d	k	pr(1)	λ	$G(D)$
0	1	0.667	0.500	$\lambda(D+D^2)$
	2	0.500	0.333	$\lambda(D+D^2+D^3)$
	3	0.400	0.250	$\lambda(D+D^2+D^3+D^4)$
	4	0.333	0.200	$\lambda(D+D^2+D^3+D^4+D^5)$
1	2	0.400	0.500	$\lambda(D^2+D^3)$
	3	0.333	0.333	$\lambda(D^2+D^3+D^4)$
	4	0.286	0.250	$\lambda(D^2+D^3+D^4+D^5)$
2	3	0.286	0.500	$\lambda(D^3+D^4)$
	4	0.250	0.333	$\lambda(D^3+D^4+D^5)$
		maxentropic		
0	1	0.724	0.618	$\lambda D+\lambda^2D^2$
	2	0.618	0.544	$\lambda D+\lambda^2D^2+\lambda^3D^3$
	3	0.566	0.519	$\lambda D+\lambda^2D^2+\lambda^3D^3+\lambda^4D^4$
	4	0.538	0.509	$\lambda D+\lambda^2D^2+\lambda^3D^3+\lambda^4D^4+\lambda^5D^5$
1	2	0.411	0.570	$\lambda^2D^2+\lambda^3D^3$
	3	0.363	0.466	$\lambda^2D^2+\lambda^3D^3+\lambda^4D^4$
	4	0.334	0.425	$\lambda^2D^2+\lambda^3D^3+\lambda^4D^4+\lambda^5D^5$
2	3	0.290	0.550	$\lambda^3D^3+\lambda^4D^4$
	4	0.262	0.430	$\lambda^3D^3+\lambda^4D^4+\lambda^5D^5$

Table B.1: Expressions for power spectra of (d,k) codes.

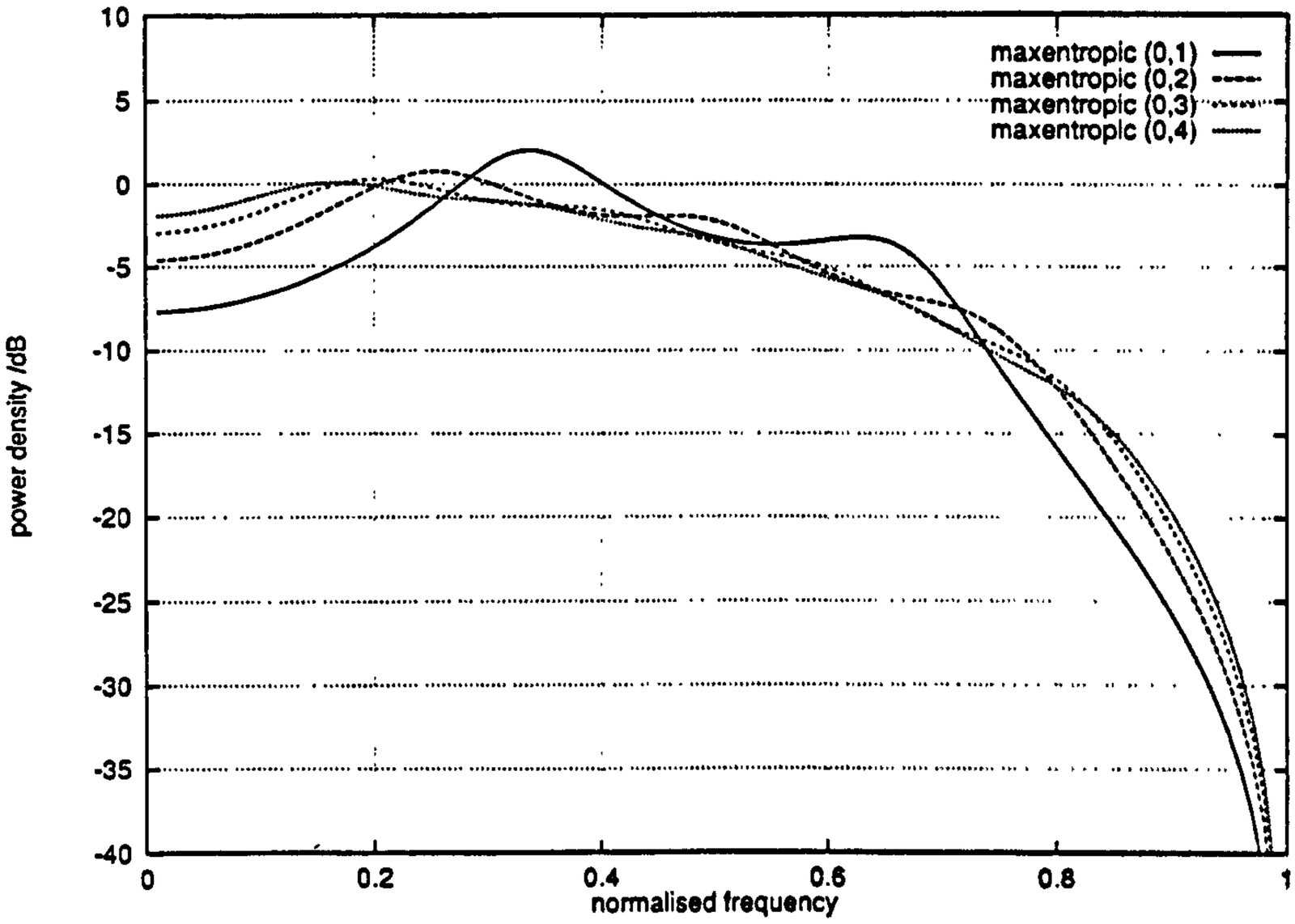


Figure B.9: Normalized maxentropic spectra for $(d, k) = (0, x)$

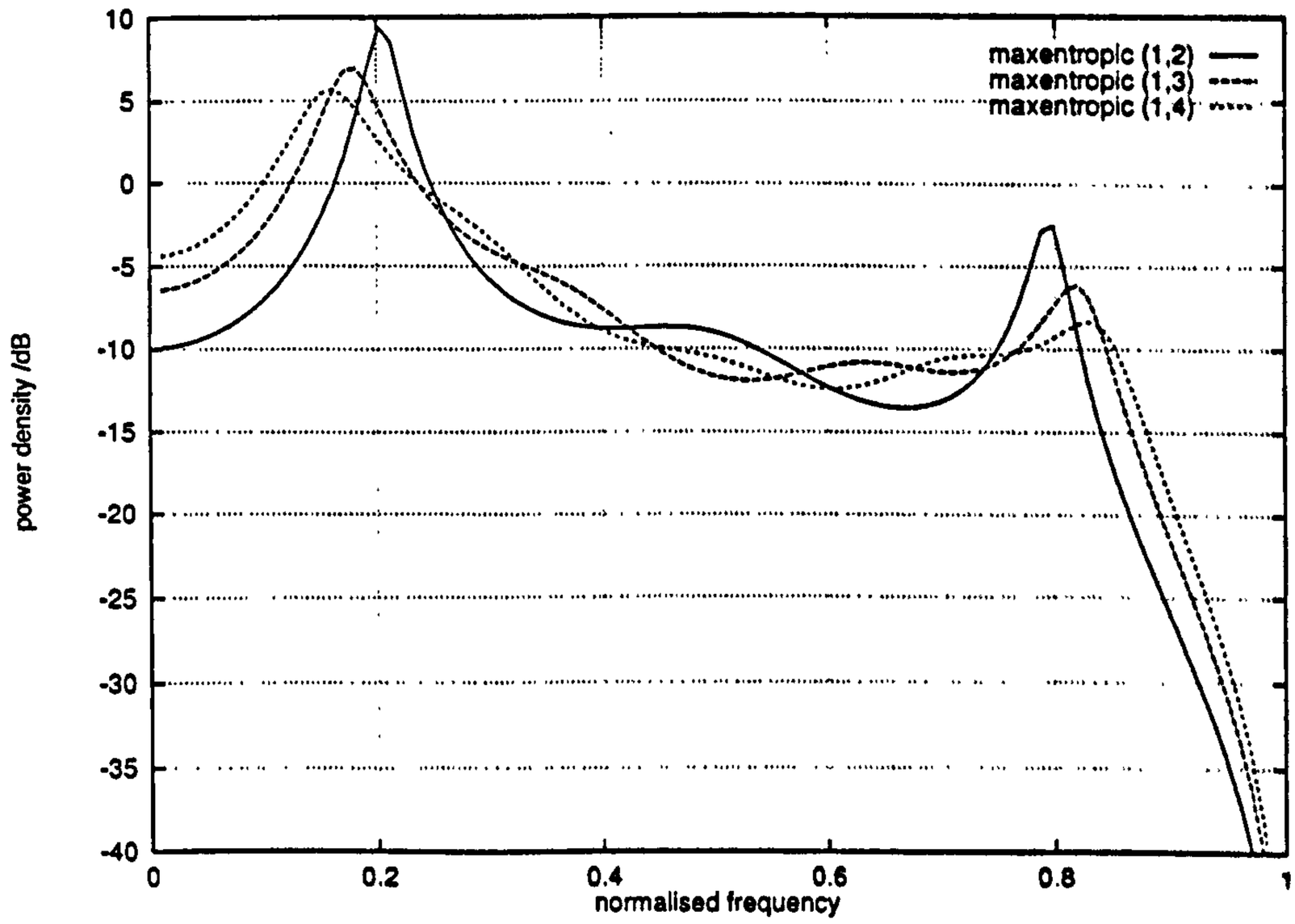


Figure B.10: Normalized maxentropic spectra for $(d, k) = (1, x)$

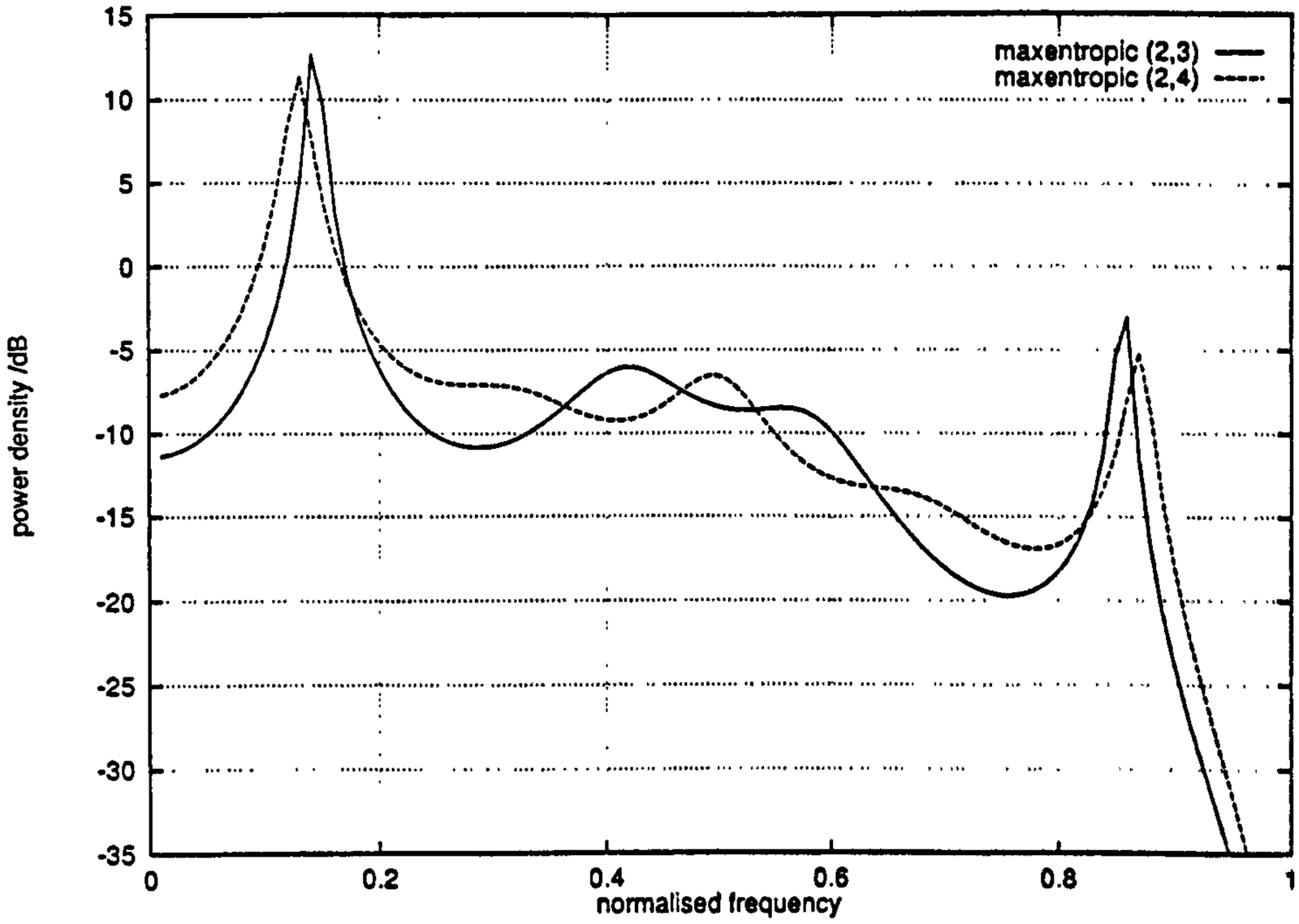


Figure B.11: Normalized maxentropic spectra for $(d, k) = (2, x)$

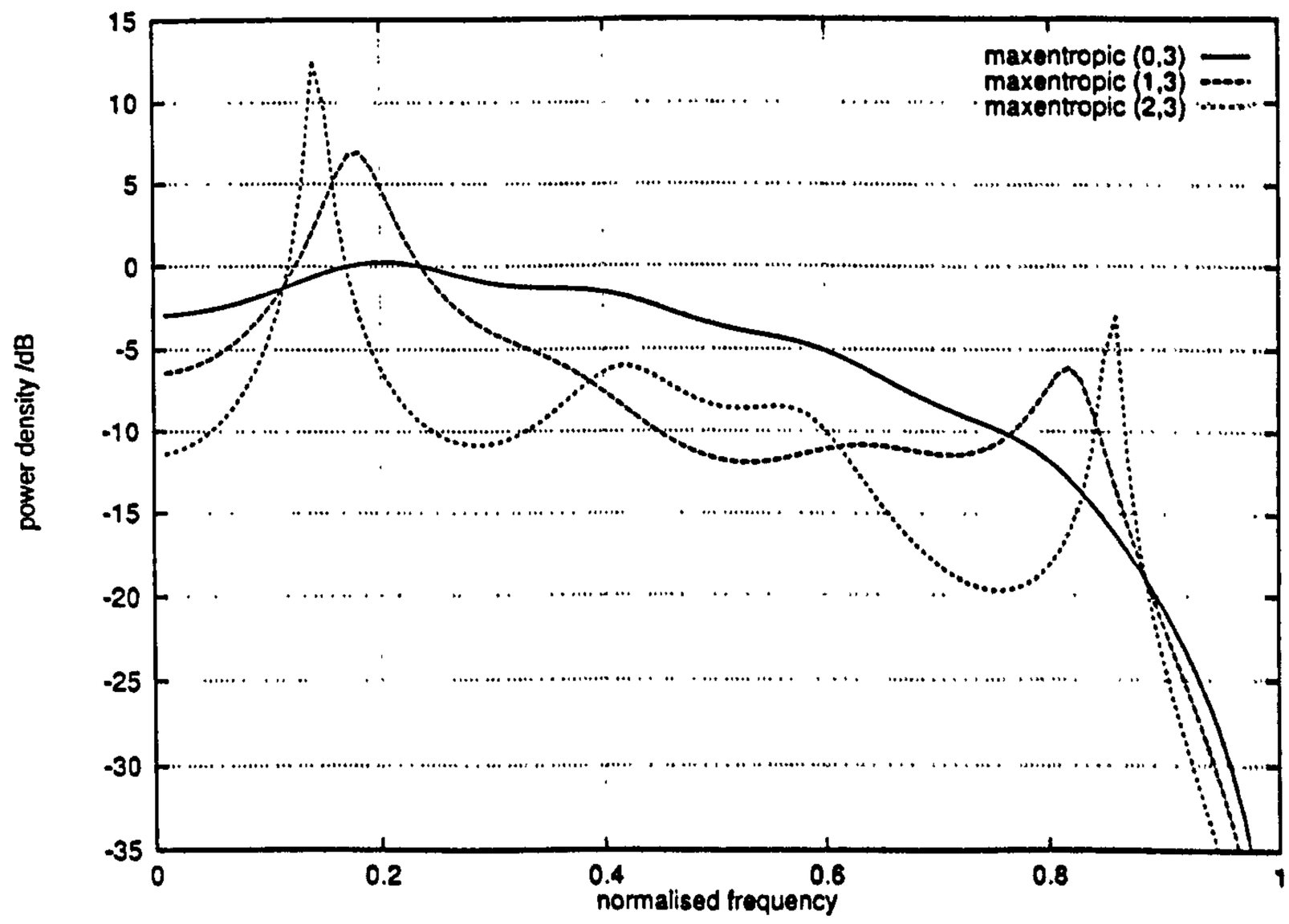


Figure B.12: Normalized maxentropic spectra for $(d, k) = (x, 3)$

Appendix C

Circuit diagrams

The following sections give a brief description of the electronic equipment specially built for this project.

C.1 Write amplifier and FIFO

The write amplifier takes an 8-bit parallel signal and strobe from the computer. This is fed into a 2 kByte FIFO and clocked out at a rate determined by a programmable clock signal generated in the computer. Another signal from the computer turns the write amplifier on or off. This amplifier will supply up to 24 mA of write current. A high power amplifier, supplying up to about 1 A of pulsed current is shown in figure 2.5 (page 44).

The following figures give the circuit diagrams for the write amplifier and FIFO board.

C.2 Read amplifier

A straightforward 8-channel differential amplifier was constructed to amplify the voltages from the read heads by about 1500. These amplifiers are suitable for both inductive and MR heads.

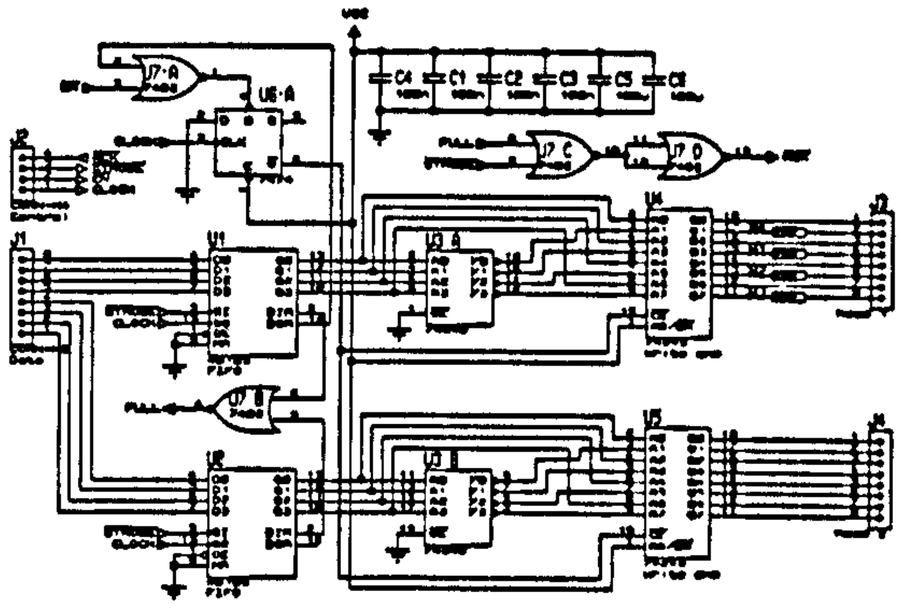


Figure C.1: Write amplifier circuit with 16-byte FIFO.

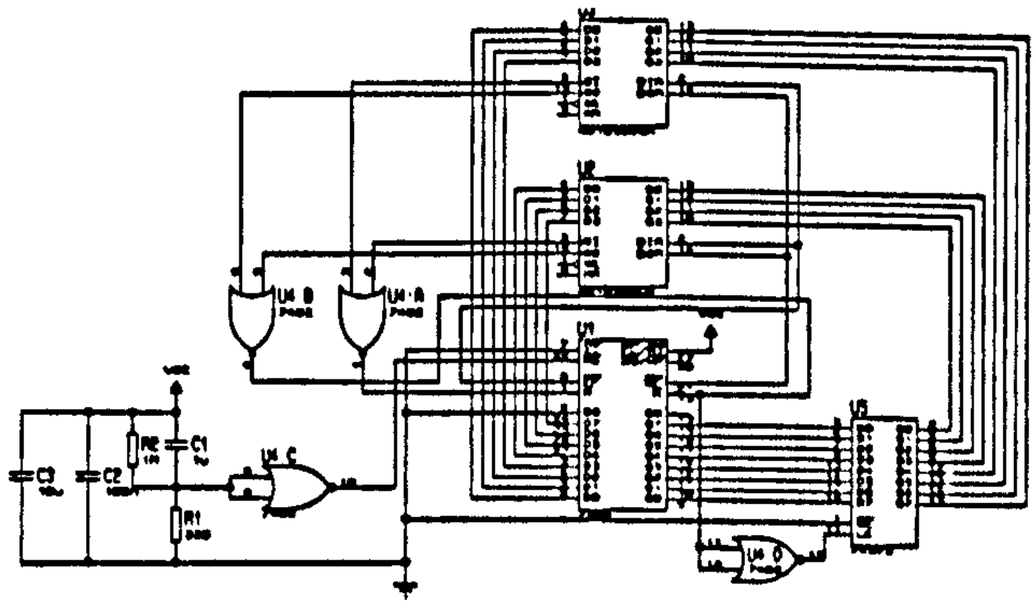


Figure C.2: FIFO expander: expands 16-byte FIFO to 2 kBytes.

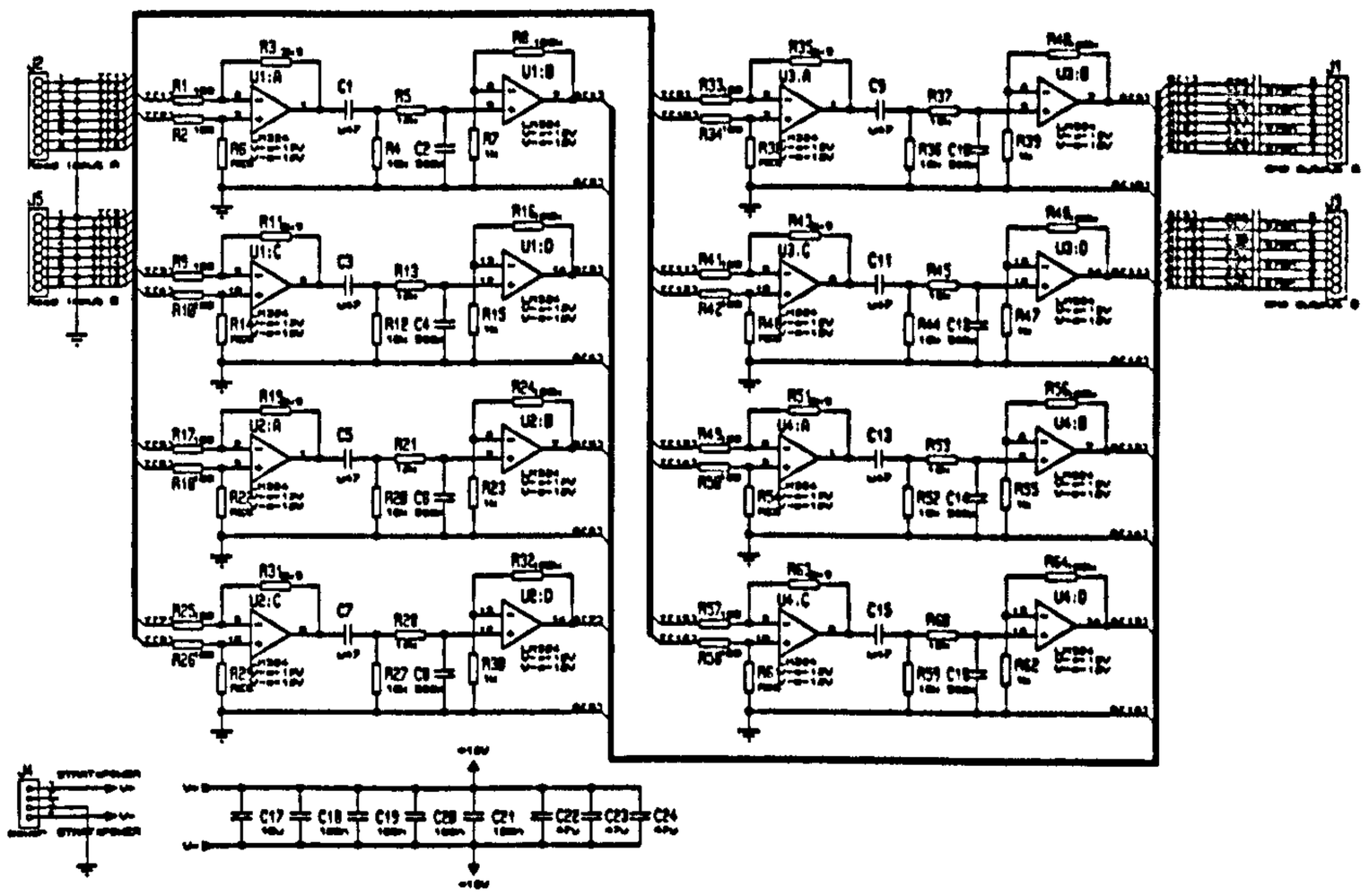


Figure C.3: Read amplifiers.

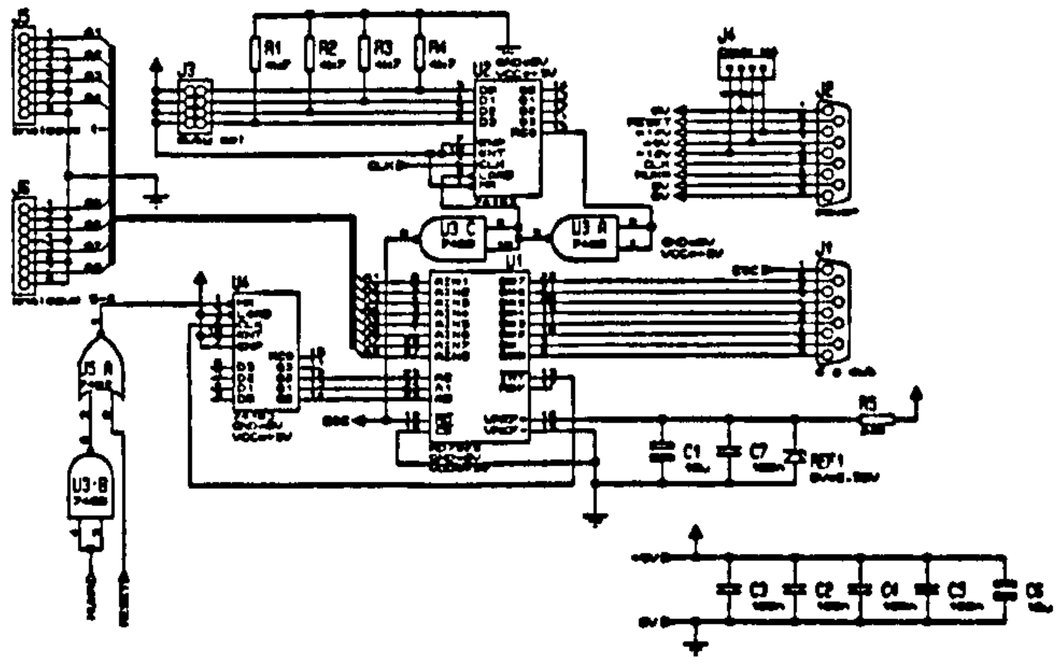


Figure C.4: Analogue to digital converter: 1–8 channels, up to 400 kHz.

C.3 Analogue to digital converter

ADC: analogue to digital converter

A single chip ADC with 8-channel track and hold is used to sample the read signals. The maximum conversion rate is about 400 kHz, shared between the number of channels used. The computer must generate a timing clock signal (which is divided to form the required synchronization signals for the ADC). In return the ADC provides an end-of-conversion signal and an 8-bit sample value. The channels are returned in order. The computer must also provide a channel-number-reset signal so that the number of sample channels is programmable.

C.4 Data transfer to the computer

Transferring 400 kBytes/s to a computer presents some technical challenges.

DMA: direct memory access

A DMA interface was designed and constructed as this gives moderately high transfer rates with minimum processor involvement. An interrupt signals when a transfer block is complete. The ISA card that was built takes an 8-bit input

ISA: Industry Standard Architecture

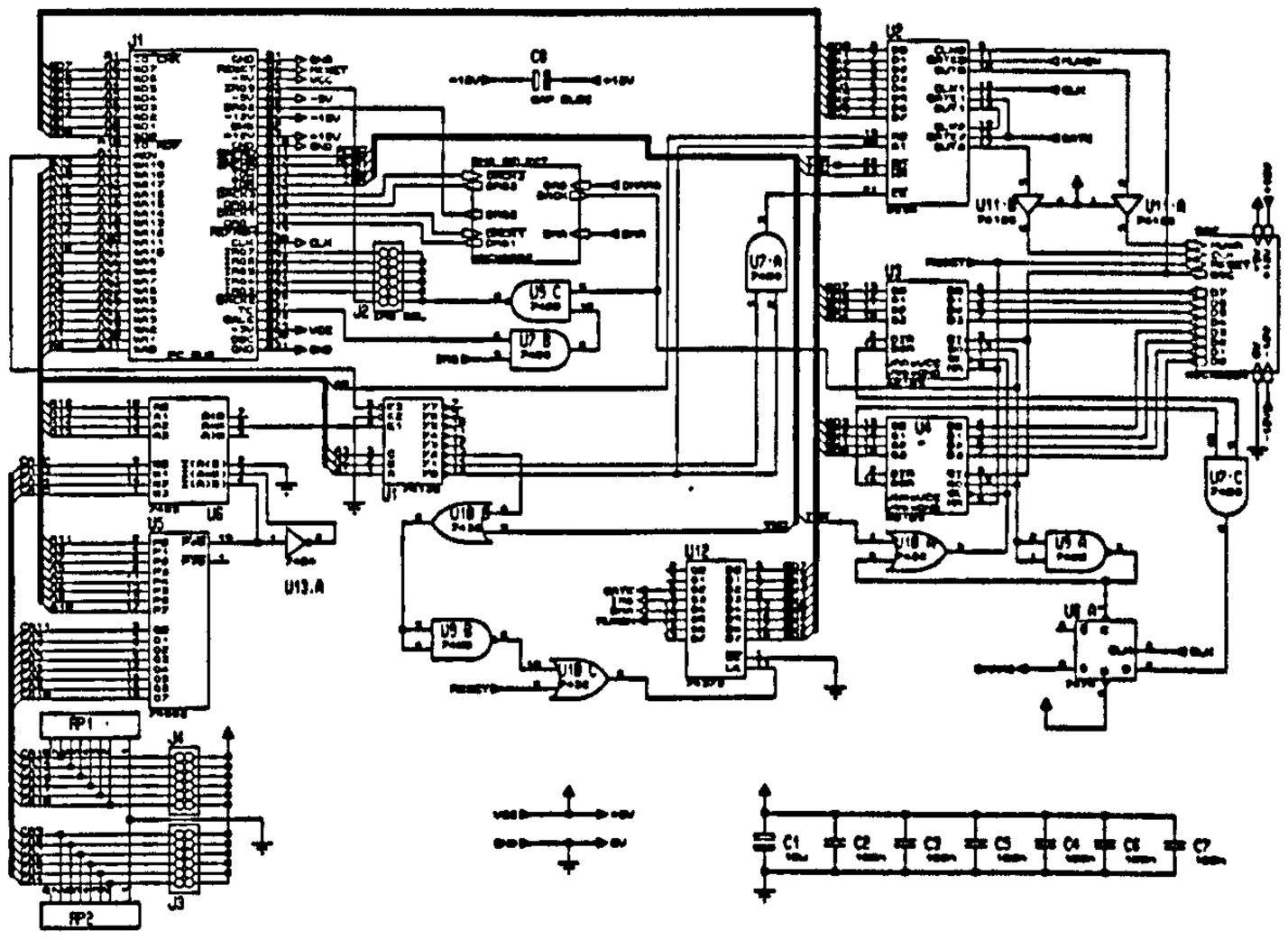


Figure C.5: Main circuit for DMA transfer ISA card.

and strobe signal and transfers this (via a small FIFO) to the DMA interface. A programmable clock and a programmable counter output are also generated.

The following figures show the ISA card circuits and DMA select jumpers and the pin layouts on the back of the card.

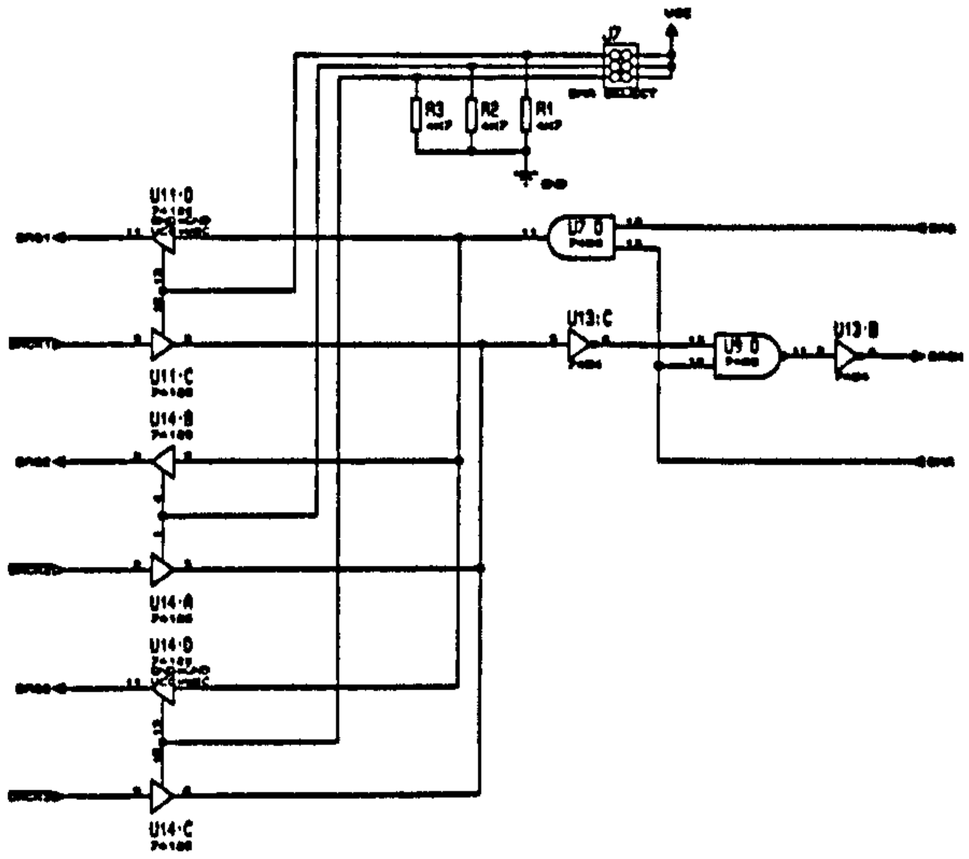


Figure C.6: DMA channel jumper select on ISA card.

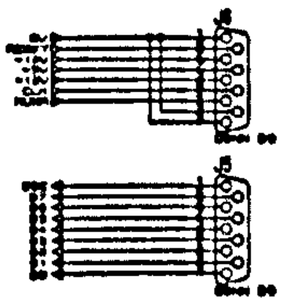


Figure C.7: External connections to DMA ISA card.

Appendix D

Software

This appendix lists some of the relevant software written in support of the project. The code is all ANSI C, and the listings here do not (intentionally) use any platform specific or special language features.

ANSI: American National Standards Institution

D.1 Variable rate code

This program reads binary data (as text '0' or '1') from the standard input and writes coded binary data (again in text form) on the standard output. An argument allows the input to be read from a file. Depending on how the program is called, it will output variable rate code with different (d, k) parameters. It is written in ANSI C and it relies on knowing the filename by which it was invoked—the (d, k) values are taken from the filename, e.g., `vr13`, `vr14`, etc. although in its present form it will only encode with $k - d = 2$ or 3.

I/O: input/output

A library of some validated I/O routines are used, and that is listed later in section D.4.

D.1.1 Encoder

```
/* > vrdk.c */
/* Daniel F. Smith, 1997 */
/* Invoked as, e.g., vr13, does variable rate coding with (d,k)=(1,3) */

#include <stdio.h>
#include <ctype.h>
#include "safer.h"
```

```
static void writen(int n) {
        /* a final stage NRZI encoder: writes out n bits, then flips sign */
        static int l=0;
        l^=1;
        for(n++;n>0;n--)
                putchar('0'+l);
        }

static void quasisym(int word,int d) {
        /* write out the quasi-symbol pair */
        int q0,q1;
        q0=word/3;
        q1=word%3;
        writen(q0+d);
        writen(q1+d);
        }

static void put2bits(int word,int d) {
        /* simply writes out 2 NRZI bits in text form */
        writen(d+word);
        }

int main(int argc,char *argv[]) {
        int word;
        int d=1,k=3;
        char *prog;
        int blocksize=3;
        void (*func)(int,int);

        prog=*(argv++),argc--;
        if (argc>0) otherstdin=*(argv++),argc--;
        if (argc!=0) return 1;

        if (prog[0]=='v' && prog[1]=='r') {
        if (isdigit(prog[2])) d=prog[2]-'0';
        if (isdigit(prog[3])) k=prog[3]-'0';
        }

switch(k-d) {
        case 2: /* use quasi-symbol method */
                blocksize=3;
                func=quasisym;
                break;
        case 3: /* simple 2 bit output */
                blocksize=2;
                func=put2bits;
                break;
        default:
                return 2;
        }

for(;;) {
        word=getnbits(blocksize);
        if (word==EOF) break;
        func(word,d);
                }
return 0;
}
```

D.1.2 Decoder

```
/* > devrdk.c */
/* Daniel F. Smith, 1997 */
/* Invoked as, e.g., devr13, does variable rate coding with (d,k)=(1,3) */
```

```
#include <stdio.h>
#include <ctype.h>
#include "safer.h"

static void quasisym(int len,int sign) {
        /* decodes quasi-symbols */
        /* needs to be called once for each q-sym with the appropriate sign */
        static int last=-1;
        if (sign!=0) {last=len; return;}
        len=last*3+len;
        last=-1;
        putchar('0'+!!(len&4));
        putchar('0'+!!(len&2));
        putchar('0'+!!(len&1));
        }

static void put2bits(int len,int sign) {
        /* decodes NRZI vr code, does not need sign */
        putchar('0'+!!(len&2));
        putchar('0'+!!(len&1));
        }

int main(int argc,char *argv[]) {
        int word,lword;
        int d=1,k=3;
        int count;
        char *prog;
        void (*func)(int,int);

        prog=*(argv++),argc--;
        if (argc>0) otherstdin=*(argv++),argc--;
        if (argc!=0) return 1;

        if (prog[0]=='d' && prog[1]=='e' && prog[2]=='v' && prog[3]=='r') {
        if (isdigit(prog[4])) d=prog[4]-'0';
        if (isdigit(prog[5])) k=prog[5]-'0';
        }

switch(k-d) {
        case 2: /* use quasi-symbol method */
                func=quasisym;
                break;
        case 3: /* simple 2 bit output */
                func=put2bits;
                break;
        default:
                return 2;
        }

count=0;
word=0;
for(;;) {
        lword=word;
        word=getnbits(1);
        if (word==EOF) break;
        if (word==lword) {count++; continue;}
        if (count>=d) func(count-d,lword);
        count=0;
        }
func(count-d,lword);
return 0;
}
```

D.2 Timing error resilience (TER)

This program calculates the timing error resilience metric (TER) from two data files of binary data (in text form). It also calculates the index of the first bit in error and the last bit in error.

```
/* > ter.c */
/* Daniel F. Smith, 1997 */
/* Given two text binary files, calculates the TER */

/*
TER (timing error resiliance) is a measure of bit errors.  E.g.,
0010010010101001001010100100101101001010010010101010101111010101101111101000
0010010110110100100101010010010110100101001001010101010111101010110111110100 0
       a^^b
    burst error with timing error
The TER for these streams would be pos(a)-pos(b)=4 errors.
*/

#include <stdio.h>
#include <stdlib.h>

static char syntax[]="<orig file> <corrupted file>";

static char *loadfile(const char *filename,unsigned long int *ext) {
        unsigned long len=0;
        FILE *f;
        char *r=NULL;
        f=fopen(filename,"r");
        if (f) {
                fseek(f,0,SEEK_END);
                len=ftell(f);
                fseek(f,0,SEEK_SET);
                r=malloc(len);
                if (r) fread(r,sizeof(char),len,f);
                fclose(f);
                }
        if (ext) *ext=len;
        return r;
        }

int main(int argc,char *argv[]) {
        char *orig,*corr; /* the files are loaded into memory here */
        unsigned long int lorig,lcorr; /* file lengths */
        int ter,up,down;

        if (argc!=3) {fprintf(stderr,"syntax: %s %s\n",argv[0],syntax); return 1;}

        orig=loadfile(argv[1],&lorig);
        corr=loadfile(argv[2],&lcorr);
        if (orig==NULL || corr==NULL) {
        fprintf(stderr,"%s: file%s not found: %s%s%s\n",argv[0],
                (!corr && !orig)?"s":"",
                orig?"":argv[1],
                (!corr && !orig)?" ":"",
                corr?"":argv[2]);
        return 2;
        }

/* remove trailing 0s (some block codes generate additional 0s) */
for(;orig[lorig-1]=='0';lorig--);
for(;corr[lcorr-1]=='0';lcorr--);

/* find first error */
for(up=0;up<lcorr && up<lorig && orig[up]==corr[up];up++);
/* and last error */
```

```
for(down=0;
    down<lcorr && down<lorig && orig[lorig-1-down]==corr[lcorr-1-down];
    down++) /* void */;

ter=lorig-down - up;
if (ter<0) ter=0; /* no errors! */
printf("%d %d %d\n",ter,up,down);
return 0;
}
```

D.3 Variable speed Bayesian detector

This is a development implementation of the variable speed Bayesian detector. It is statically compiled to look `FORW` bits ahead in time with `BACK` bits of decision feedback with variable rate $(d, k) = (1, 3)$. The intermediate results are displayed as several dynamic charts in a window by the functions in the file `scopewin` (which is not listed here).

The program is invoked simply with the initial oversampling rate. The readback signal is read from the standard input as unsigned byte data. The output is the binary data (in text form) and the standard error output gives a reading of current decoding statistics.

A library of some validated I/O routines are used, and that is listed in section D.4.

D.3.1 Main loop

```
/* > bayes.c */
/* Daniel F. Smith, 1997 */
/* Implementation of blind Bayesian decoder */
/* With variable rate code discrimination */
/* With speed tracking */

/* control whether to display charts on screen */
#define SCOPE
/* implement Bayesian detector or FDTS detector */
#define BAYES

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "safer.h"
#include "filter.h"
#include "vrcode.h"
#include "sigex.h"
#ifdef SCOPE
#include "scopewin.h"
#endif
```

```
#define FORW 4          /* forward check bits (look ahead) */
#define BACK 17         /* reverse check bits (decision feedback) */
#define TOTAL (FORW+1+BACK)
#define OVERSAMPLE 4    /* internal oversampling rate */
#define SQUARE(a) ((a)*(a))
#define BITTOD(a) ((a)?127.0:-128.0)
const double alpha=1.0; /* controls variable speed (1.0) or phase (0.0) */
double mub2=1e-6/256;   /* LMS (mu/2)/256 */

typedef unsigned char uchar;

typedef struct {
        double Z,X,Y;
        double epsilon,delta;
        } stats;        /* statistics of snatch buffers */

static double subtract(double a,double b) {
        /* threshold subtractor: FSD on the input is +/-128 */
        const double t=80.0;
        if (a>+t && b>+t) return 0.0;
        if (a<-t && b<-t) return 0.0;
        return a-b;
        }

static double streamexp(bitstream b,int index) {
        /* reads a sample from a bitstream object */
        index/=OVERSAMPLE;
        if (index>TOTAL) return 0.0;
        return BITTOD(b & bit(TOTAL-1-index));
        }

static void showscope(filter *est,filter *real,filter *chan,bitstream b) {
        /* display charts on screen */
        #ifdef SCOPE
        static scopet *s=NULL;
        int i;
        if (!s) s=newscope(5);
        for(i=0;i<est->ntaps;i++) {
                double c;
                c=(i>=chan->ntaps)?0.0:chan->tap[i];
                scope(s,c*1200,streamexp(b,i)/4.0,
                        est->tap[i],real->tap[i],
                        subtract(est->tap[i],real->tap[i])/2.0);
                }
        scopeblip(s);
        #endif
        }

static filter *makeest(filter *dest,bitstream b,const filter *channel) {
        /* convert a bitstream into a snatch */
        int i;
        for(i=0;i<dest->ntaps;i++)
                dest->tap[i]=streamexp(b,i);
        dest->scale=1.0/256.0;
        filfirf(dest,channel);
        return dest;
        }

static double compare(filter *est,filter *real) {
        /* calculate the probability that the real filter was caused by est */
        double sumsq=0.0;
        int i;
        for(i=0;i<est->ntaps;i++) {
                double d=subtract(est->tap[i],real->tap[i]);
                sumsq+=d*d;
                }
        sumsq/=256*OVERSAMPLE*TOTAL;
        return exp(-sumsq);
        }
```

```
static stats *update(stats *result,
        const filter *est,const filter *real,const filter *channel,
        bitstream known,double expd) {
        /* calculate snatch statistics and update the LMS filter */
        double integ=0.0;         /* LHS */
        double ss=0.0,ssi=0.0;    /* both RHSs */
        double d;                 /* d(real)/dt */
        double e;                 /* d(est)/dt */
        double f;                 /* average of d and e */
        int i,m;
        double x;

        for(i=1;i<real->ntaps-1;i++) {
                d=0.5*(real->tap[i+1]-real->tap[i-1]);
                e=0.5*( est->tap[i+1]- est->tap[i-1]);
                f=0.5*(d+e);
                integ+=subtract(est->tap[i],real->tap[i])*f;
                f*=f;
                ss+=f;
                ssi+=i*f;
                }
        result->Z=integ;
        result->X=ss/expd;
        result->Y=ssi;
        result->delta  =1.00*result->Z/result->Y;
        result->epsilon=0.00*result->Z/result->X;

        /* LMS filter update */
        m=(channel->ntaps-OVERSAMPLE)/2;
        for(i=0;i<OVERSAMPLE;i++) {
                int j;
                x=mub2*subtract(real->tap[m+i],est->tap[m+i]);
                for(j=0;j<channel->ntaps;j++)
                        channel->tap[j]+=x*streamexp(known,i+j-OVERSAMPLE/2);
                }

        return result;
        }

int main(int argc,char *argv[]) {
        int bestno=-1; /* index number of the best match so far */
        bitstream bestknown=nullstream; /* the best match so far */
        bitstream known=0; /* decision feedback bits */
        uchar *buf; /* the data buffer */
        double boff; /* buffer offset */
        long int len; /* length of buffer */
        double expd=2.0; /* estimated expansion factor */
        stats st; /* a holder for snatch statistics */
        double eta,maxeta,sgneta; /* probability calculations */
        filter *r; /* the real data, expanded to OVERSAMPLE */
        filter *rhat; /* the estimated data from a bitstream, expanded */
        filter *channel; /* the estimated channel response, expanded */
        int i;

        if (argc==2) expd=atof(argv[1]); /* starting expansion factor */
        len=readstream(stdin,(char **)&buf); /* will fail if out of memory */
        boff=0.5;

        r=filnew("resampled data",OVERSAMPLE*TOTAL);
        rhat=filnew("estimated data",OVERSAMPLE*TOTAL);
        channel=filnew("channel estimate",OVERSAMPLE*BACK);

        /* start off with a channel estimate: impulse or dipulse */
        for(i=0;i<channel->ntaps;i++) {
double a=i-(channel->ntaps/2);
                /* channel->tap[i]=.1/(1+0.10*a*a); */ /* impulse/Lorentzian */
                channel->tap[i]=(0.02*a)/SQUARE(1+0.02*a*a); /* dipulse */
                }
```

```
while((len-boff)>expd*TOTAL) {
        /* generate real sampled values */
        sigexpandfromc(buf,len,boff,TOTAL*expd,r->tap,r->ntaps);
        /* generate streams */
        maxeta=-1.0; /* always update */
        sgneta=0.0;
        for(i=0;i<(1<<(FORW+1));i++) {
                /* work through every possible signal */
                /* this doesn't calculate eta(+1) and eta(-1) but
                 * adds or subtracts eta() depending on what s_0
                 * is---the cumulation is in sgneta. */
                bitstream d;
                d=adddk13code(known,FORW+1,i);
                if (d==nullstream) continue;
                /* turn into a signal estimate */
                makeest(rhat,d,channel);
                /* compare */
                eta=compare(rhat,r);
                if (eta>maxeta) /* it's more probable than others */ {
                        bestno=i;
                        maxeta=eta;
                        bestknown=d;
                        }
                if (d & bit(FORW)) eta=-eta;
                sgneta+=eta; /* -ve for 1, +ve for 0 */
                }
/* update with FDTS estimate */
known=bestknown;
#ifdef BAYES
/* convery FDTS result into Bayes form */
if (sgneta<0.0) known|= bit(FORW);
if (sgneta>0.0) known&=~bit(FORW);
#endif

/* calculate statistics for LMS update */
makeest(rhat,known,channel);
update(&st,rhat,r,channel,known,expd);

/* show stuff on screen */
showscope(rhat,r,channel,known);

/* print useful values */
fprintf(stderr,"pos %.1f sgneta %.3e maxeta %.3e "
                "expd %.4f delta %.4f\n",
                 boff,sgneta,maxeta,expd,st.delta);

/* output the decoded data */
known>>=FORW;
putchar('0'+(known&1)); fflush(stdout);

/* update the variable speed tracking */
expd*=(1+alpha*st.delta);
if (expd< 1.0) expd= 1.0; /* this is useful for real data */
if (expd>20.0) expd=20.0;
boff+=expd-st.epsilon;
   }
return 0;
}
```

D.3.2 Support routines: variable rate code

```
/* > vrcode.h */
#ifndef VRCODE_H
#define VRCODE_H
```

```
#define shl(x,y) ((x)<<(y))
#define shr(x,y) ((x)>>(y))
#define bit(x) (1<<(x))
#define rbit(a,x) (!!((a) & bit(x)))
#define ones(a) (bit(a)-1)

/* holds a transmitted bit stream, most recent in LSB */
typedef unsigned long int bitstream;
extern const bitstream nullstream;
/* all bitstreams are valid, but nullstream is special in some circumstances */

extern int qcode[],s0,nsa,nsb;
extern char *qsym[];
extern bitstream getvrcode(int n,int l);
extern bitstream adddk13code(bitstream prev,int n,int index);
extern char *bitstreamtoa(bitstream);
#endif



/* vrcode.c */
/* Daniel F. Smith, 1996 */
/* contains the code parameters */
#include "vrcode.h"
#include "safer.h"
const bitstream nullstream=~(bitstream)0;

/* quasi-symbol mappings, S_a in high nybble, S_b in low nybble */
int  qcode[9]={0x00, 0x01, 0x02, 0x10, 0x11, 0x12, 0x20, 0x21, 0x22};
/* maps symbols back to quasi-symbols, qsym[nsb*S_a+S_b] */
char *qsym[9]={"000","001","010","011","100","101","110","111","SSS"};
int s0=2;
int nsa=3,nsb=3;

static bitstream genvrcode(int b,int l) {
        /* generate variable rate code coding b */
        /* maximum length of code is l */
        /* first bits are always 0 */
        /* bit 0 of b is always aligned at bit l-1 of answer */
        bitstream r=0;
        int i=0;
        while(i<l) {
                int x,p,q;
                x=b & 7;
                b>>=3;
                p=s0+(qcode[x]>>4);
                q=s0+(qcode[x]&15);
                r<<=p; i+=p; r|=0;
                r<<=q; i+=q; r|=ones(q);
                }
        if (i>l) r>>=(i-l);
        return r;
        }

bitstream getvrcode(int n,int l) {
        /* gets an unique VR code of index n */
static int ll=0;
static int bn=0;
static bitstream *code=(void *)0;
static int tablen=0;
if (l!=ll) /* regenerate table */ {
        int t,i,j;
        t=1<<(l/2+1);
        if (t>tablen) {
                tablen=t+20;
                code=srealloc(code,tablen*sizeof(*code));
                }
        bn=0;
        for(i=0;i<t;i++) {
                bitstream x;
```

```
                        x=genvrcode(i,l);
                        for(j=0;j<bn;j++)
                                if (x==code[j]) break;
                        if (j==bn) code[bn++]=x;
                        }
                code[bn]=0;
                ll=l;
                }
        if (n>=bn) return 0;
        return code[n];
        }

bitstream adddk13code(bitstream prev,int n,int index) {
        /* add n bits onto prev (recursive) */
        /* index determines what value to add */
        /* nullstream returned if index is not a valid expansion */
        bitstream temp=prev;

        if (n<=0) {
                if (index!=0) return nullstream;
                return prev;
                }
        prev<<=1;
        n--;

        /* k constraint */
        temp&=ones(4); /* can only flip state */
        if (temp==0      ) return adddk13code(prev|1,n,index);
        if (temp==ones(4)) return adddk13code(prev|0,n,index);

        /* d constraint */
        temp&=ones(2); /* can't be the same as prev bit */
        if (temp==2) return adddk13code(prev|0,n,index);
        if (temp==1) return adddk13code(prev|1,n,index);

        /* take our index code out */
        return adddk13code(prev|(index&1),n,index/2);
        }

char *bitstreamtoa(bitstream x) {
        /* convert a bitstream into text form */
        const int sz=sizeof(bitstream)*8;
        static char buf[sz+1];
        int i=sz;
        buf[i--]='\0';
        while(i>=0) {
                buf[i--]='0'+(x&1);
                x>>=1;
                }
        return buf;
        }
```

D.3.3 Support routines: signal expansion

```
/* > sigex.h */
#ifndef sigex_h
#define sigex_h

extern int sigexpand(const double *in,int inlen,double *out,int outlen);
extern int sigexpandc(const unsigned char *in,int inlen,double *out,int outlen);
extern int sigexpandsqu(const double *in,int inlen,double *out,int outlen);
extern int sigexpandlin(const double *in,int inlen,double *out,int outlen);
extern int sigexpandfromc(const unsigned char *buf,int buflen,
                        double off,double len,double *out,int outlen);

#endif
```

```
/* > sigex.c */
/* Daniel F. Smith, 1996 */
/* squeezes or expands a block of (double) data */
#include <stdlib.h>
#include "safer.h"
#include "qsinc.h"

static int squash(const double *in,int inlen,double *out,int outlen,double sc) {
        int i,j,j1,j2,jd,t;
        double p=(double)(inlen-1)/(outlen-1);
        double ex=qsinc_ext();
        jd=ex/sc+0.5;
        for(i=0;i<outlen;i++) {
                double s=0.0;
                t=p*i+0.5;
                j1=t-jd; if (j1<0) j1=0;
                j2=t+jd; if (j2>inlen) j2=inlen;
                for(j=j1;j<j2;j++)
                        s+=qsinc(sc*(j-t))*in[j];
                out[i]=s/p;
                }
        return 0;
        }

static int expand(const double *in,int inlen,double *out,int outlen) {
        int i,f,r;
        int newinlen;
        double *b;
        double s;
        f=outlen/inlen;
        if (inlen*f!=outlen) f++;
        /* expand up a multiple */
        newinlen=1+f*(inlen-1);
        b=(double *)calloc(newinlen,sizeof(double));
        if (!b) return 1;
        s=(double)newinlen/outlen;
        for(i=0;i<inlen;i++)
                b[f*i]=s*in[i];
        r=squash(b,newinlen,out,outlen,PI/f);
        free(b);
        return r;
        }

static int copy(const double *in,double *out,int len) {
        int i;
        for(i=0;i<len;i++)
                out[i]=in[i];
        return 0;
        }

int sigexpand(const double *in,int inlen,double *out,int outlen) {
        /* 'perfect' expand: convolve with sinc */
        if (!out || !in) return 1;
        if (inlen==outlen) return copy(in,out,outlen);
        if (inlen>outlen) return squash(in,inlen,out,outlen,(PI*outlen-1)/(inlen-1));
        return expand(in,inlen,out,outlen);
        }

int sigexpandc(const unsigned char *in,int inlen,double *out,int outlen) {
        /* expand from unsigned characters */
        static double *buf=NULL;
        static int buflen=0;
        int i;
        if (inlen>buflen) {
                buflen=inlen+20;
                buf=srealloc(buf,buflen*sizeof(*buf));
                }
        for(i=0;i<inlen;i++)
                buf[i]=uctod(in[i]);
        return sigexpand(buf,inlen,out,outlen);
```

```
        }

static double getpelc(const unsigned char *buf,int buflen,double x,double ex) {
        /* return the interpolated value at (arbitrary point) x of buf */
        double sum=0.0;
        int i,p1,p2;
        p1=x-ex+1.0;
        p2=x+ex;
        if (p1<0) p1=0;
        if (p2>buflen) p2=buflen;
        for(i=p1;i<p2;i++)
                sum+=qsinc(PI*(x-i))*(buf[i]-128.0);
        return sum;
        }

int sigexpandfromc(const unsigned char *buf,int buflen,
                        double off,double len,
                        double *out,int outlen) {
        /* expand from a char array from arbitrary position and length */
        int i;
        double step,ex;
        ex=qsinc_ext()/PI; /* pixel extent */
        step=len/(outlen);
        for(i=0;i<outlen;i++)
                out[i]=getpelc(buf,buflen,off+i*step,ex);
        return 0;
        }

int sigexpandsqu(const double *in,int inlen,double *out,int outlen) {
        /* unsophisticated block expand with interpolation at edges */
        double sc,off;
        int i,j,k;
        if (!out || !in) return 1;
        if (inlen==outlen) return copy(in,out,outlen);
        sc=(double)inlen/outlen;
        for(i=0;i<outlen;i++) {
                j=sc*i;
                k=sc*(i+1);
                if (j==k || k>=inlen) {
                        out[i]=in[j];
                        continue;
                        }
                off=sc*i-j;
                out[i]=in[j]*(1.0-off)+in[k]*off;
                }
        return 0;
        }

int sigexpandlin(const double *in,int inlen,double *out,int outlen) {
        /* simple linear interpolation */
        double sc,off;
        int i,j;
        if (!out || !in) return 1;
        if (inlen==outlen) return copy(in,out,outlen);
        sc=(double)(inlen-1)/(outlen-1);
        for(i=0;i<outlen;i++) {
        j=sc*i;
        off=sc*i-j;
        out[i]=(1.0-off)*in[j];
        if ((++j)<outlen) out[i]+=off*in[j];
        }
return 0;
}
```

D.3.4 Support routines: filtering

```
/* > filter.h */
#ifndef filter_h
#define filter_h
#include <stdio.h>

typedef double ftype;

typedef struct {
        char *name;    /* a text name for the filter */
        ftype *tap;    /* the filter taps */
        int ntaps;     /* the number of filter taps */
        double scale; /* a linear amplitude scale factor */
        } filter;

extern filter *filload(const char *);
extern filter *filfload(const char *,FILE *f);
extern filter *filcopy(const filter *);
extern filter *filnew(const char *,int);
extern void filresize(filter *,int);
extern void fildel(filter *);
extern void filadd(filter *,const filter *);
extern void filfiri(unsigned char *,int,const filter *);
extern void filfir(double *,int,const filter *);
extern void filfirf(filter *,const filter *);

#endif



/* > filter.c */
/* Daniel F. Smith, 1996 */
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "safer.h"
#include "filter.h"

filter *filnew(const char *name,int ntaps) {
        /* create a new filter */
        filter *n;
        n=salloc(1,filter);
        n->name=salloc(strlen(name)+1,char);
        strcpy(n->name,name);
        n->tap=scalloc(ntaps,sizeof(*(n->tap)));
        n->ntaps=ntaps;
        n->scale=1.0;
        return n;
        }

void filresize(filter *f,int ntaps) {
        /* change the number of taps in a filter (no initialization) */
        f->tap=srealloc(f->tap,ntaps*sizeof(*(f->tap)));
        f->ntaps=ntaps;
        }

filter *filcopy(const filter *f) {
        /* create a copy of a filter */
        filter *n;
        int i;
        if (!f) return NULL;
        n=filnew(f->name,f->ntaps);
        for(i=0;i<f->ntaps;i++)
                n->tap[i]=f->tap[i];
        return n;
        }

void fildel(filter *f) {
        /* delete a filter */
```

```
        free(f->name);
        free(f->tap);
        free(f);
        }

filter *filload(const char *name) {
        /* load a filter from filename */
        filter *fil;
        FILE *f;
        f=fopen(name,"r");
        fil=filfload(name,f);
        fclose(f);
        return fil;
        }

filter *filfload(const char *name,FILE *f) {
        /* load a filter from FILE* */
        char tmp[40];
        filter *fil;
        int i;

        fil=filnew(name,512);
        for(i=0;!feof(f);) {
                sfgets(tmp,lengthof(tmp),f);
                if (tmp[0]=='#') continue;
                if (sscanf(tmp,"%lf",&(fil->tap[i++]))!=1) break;
                if (i>=fil->ntaps) filresize(fil,2*fil->ntaps);
                }
        filresize(fil,i-1);
        return fil;
        }

void filadd(filter *d,const filter *s) {
        /* add two filters together */
        int i;
        double sc=1.0;
        if (d->ntaps != s->ntaps) return;
        if (d->scale!=0.0) sc=s->scale/d->scale;
        for(i=0;i<d->ntaps;i++)
                d->tap[i]+=sc*s->tap[i];
        }

/* The following routines perform FIR on different types of input data */

void filfiri(unsigned char *id,int len,const filter *f) {
        int a,b;
        double mac;
        static unsigned char *idd=NULL;
        static int iddlen=-1;

        if (!f || !id || len<=0) return;
        if (iddlen!=len) {
                iddlen=len;
                idd=(unsigned char *)srealloc(idd,iddlen*sizeof(unsigned char));
                }
for(a=0;a<len;a++)
        idd[a]=id[a];

for(a=0;a<len;a++) {
        mac=0.0;
        for(b=0;b<f->ntaps;b++) {
                int from;
                from=a+b-(f->ntaps)/2;
                if (from<0 || from>len) continue;
                mac+=(idd[from]-128)*f->tap[b];
                }
        mac*=f->scale;
        mac+=128;
        if (mac>255) mac=255;
        if (mac<0) mac=0;
```

```
                id[a]=mac;
                }
        }

void filfir(double *id,int len,const filter *f) {
        int a,b;
        double mac;
        static double *idd=NULL;
        static int iddlen=-1;

        if (!f || !id || len<=0) return;
        if (iddlen<len) {
                iddlen=len+20;
                idd=(double *)srealloc(idd,iddlen*sizeof(double));
                }
        for(a=0;a<len;a++)
                idd[a]=id[a];

        for(a=0;a<len;a++) {
                mac=0.0;
                for(b=0;b<f->ntaps;b++) {
                        int from;
                        from=a+b-(f->ntaps)/2;
                        if (from<0 || from>len) continue;
                        mac+=idd[from]*f->tap[b];
                        }
                mac*=f->scale;
                id[a]=mac;
                }
        }

void filfirf(filter *to,const filter *by) {
        int a,b,x,y,z;
        double mac;
        static filter *src=NULL;

        src=filcopy(to);
        if (to==by) by=src; /* src won't be altered---safe for autocorrel. */

        for(a=0;a<to->ntaps;a++) {
                x=a-(by->ntaps)/2;          /* from in src */
                y=x+by->ntaps;              /* to   in src */
                z=0;                        /* start in by */
                if (x<0) z-=x,x=0;
                if (y>src->ntaps) y=src->ntaps;
                mac=0.0;
                for(b=x;b<y;b++,z++)
                        mac+=src->tap[b]*by->tap[z];
                mac*=by->scale;
                to->tap[a]=mac;
                }
        fildel(src);
        }
```

D.3.5 Support routines: quick sinc function

```
/* > qsinc.h */
#ifndef qsinc_h
#define qsinc_h

extern int qsinc_init(int,double);
extern double qsinc(double);
extern double qsinc_ext(void);

#endif
```

```
#ifndef PI
#define PI (3.141592653589793 23)
#endif



/* > qsinc.c */
/* Daniel F. Smith, 1996 */
/* Fast sin(x)/x */
/* uses truncated lookup table and linear interpolation */
#include <math.h>
#include <stdlib.h>
#include "qsinc.h"

static double *buf=NULL;
static int buflen=0;
static double sc=0.0;

#define QSINCDEF qsinc_init(20,4*PI)

static double sinc(double x) {
        /* probably won't do very well close to x=0 */
        /* you should use such large tables in that case */
        if (x==0.0) return 1.0;
        return sin(x)/x;
        }

static double window(double x) {
        /* use Welch window for truncation */
        /* -1<x<1 */
        return 1-x*x;
        }

int qsinc_init(int size,double extent) {
        /* size=size of lookup table */
        /* extent=truncation extent */
        int i;
        if (size<=0) return 1;
        if (buf) free(buf);
        buf=(double *)malloc(sizeof(double)*size);
        if (!buf) return 2;
        buflen=size;
        sc=(double)buflen/extent;
        for(i=0;i<buflen;i++)
                buf[i]=sinc((i*extent)/buflen)*window((double)i/buflen);
        return 0;
        }

double qsinc(double x) {
        int i;
        double off,add1,add2=0.0;
        if (!buf && QSINCDEF) return 0.0;
        if (x<0.0) x=-x;
        x*=sc;
        i=x;
        if (i>=buflen) return 0.0;
        off=x-i;
        add1=(1.0-off)*buf[i];
        if (++i<buflen) add2=off*buf[i];
        return add1+add2;
        }

double qsinc_ext(void) {
        if (!buf && QSINCDEF) return 0.0;
        return buflen/sc;
        }
```

D.4 Support library

This library provides simple validated routines that are generally useful.

```
/* > safer.h */
#ifndef safer_h
#define safer_h
#include <stdio.h>

#define lengthof(x) (sizeof(x)/sizeof(*(x)))
#define salloc(n,t) scalloc((n),sizeof(t))
#define square(x) ((x)*(x))

extern char *otherstdin;

extern double uctod(unsigned char);
extern unsigned char dtouc(double);
extern char dtoc(double);
extern void *scalloc(size_t,size_t);
extern void *srealloc(void *,size_t);
extern int sfgets(char *,int,FILE *);
extern size_t readstream(FILE *,char **);
extern char *itob(unsigned int);
extern FILE *sfopen(const char *,const char *);
extern int bitrev(int,int);
extern int getnextbit(void);
extern int getnbits(int);
extern int getnnrzbits(int);
extern int countones(unsigned int);
extern void outputnrz(int);
extern void outputnrzbits(int,unsigned int);
extern double fact(int);
#endif
```

```
/* > safer.c */
/* Daniel F. Smith, 1996 */
#include <stdlib.h>
#include <stdio.h>
#include "safer.h"

char *otherstdin=NULL; /* if non-null, get*bit (*!=next) reads from this */

unsigned char dtouc(double d) {
        /* double to unsigned char */
        int x=d+128;
        if (x>255) x=255;
        if (x<0) x=0;
        return x;
        }

char dtoc(double d) {
        /* double to unsigned char with no offset */
        int x=d;
        if (x>255) x=255;
        if (x<0) x=0;
        return x;
        }

double uctod(unsigned char c) {
        /* unsigned char to double */
        int x=c;
        return (double)(x-128);
        }

FILE *sfopen(const char *name,const char *mode) {
        /* safer file open */
```

```
        FILE *r;
        r=fopen(name,mode);
        if (r) return r;
        perror(name);
        exit(1);
        }

/* safer versions of memory allocation */

void *scalloc(size_t n,size_t size) {
        void *r;
        r=calloc(n,size);
        if (r) return r;
        fprintf(stderr,"Out of memory allocating %d bytes\n",n*size);
        exit(1);
        }

void *srealloc(void *m,size_t x) {
        m=realloc(m,x);
        if (m) return m;
        fprintf(stderr,"Out of memory allocating %d bytes\n",x);
        exit(1);
        }

int sfgets(char *buf,int size,FILE *f) {
        /* more sensible version of fgets() */
        int i;
        if (!f) return 0;
        for(i=0;i<size-1 && !feof(f);i++) {
                int c;
                c=fgetc(f);
                if (c==EOF || c=='\n' || c=='\r' || c=='\0') break;
                buf[i]=c;
                }
        buf[i++]='\0';
        return i;
        }

size_t readstream(FILE *f,char **h) {
        /* reads a non-regular file into memory */
        size_t len=1024,sofar=0,nextread=len;
        char *buf;
        if (!h) return 0;
        buf=srealloc(NULL,len);
        do { /* almost got to do it this way unfortunately */
                unsigned long int l;
                l=fread(buf+sofar,sizeof(char),nextread,f);
                sofar+=l*sizeof(char);
                if (l<nextread) break;
                len*=2;
                nextread=len-sofar;
                buf=srealloc(buf,len);
                } while(!feof(f));
        *h=buf;
        return sofar;
        }

char *itob(unsigned int x) {
        /* integer to binary (text) */
        static char buf[1+8*sizeof(unsigned int)];
        int i;
        for(i=sizeof(buf)-1;i>0;x>>=1)
                buf[--i]='0'+(x&1);
        buf[sizeof(buf)-1]='\0';
        return buf;
        }

int countones(unsigned int x) {
        /* count the number of set bits in x */
        int i;
```

```
        for(i=0;x;i++)
                x&=x-1;
        return i;
        }

static int rev(int x,int l) {
        /* slowly reverse the bits in a number */
        int y=0;
        int i;
        for(i=l;i>0;i--) {
                y<<=1;
                y|=x&1;
                x>>=1;
                }
        return y;
        }

int bitrev(int x,int l) {
        /* fast (lookup table) bit reverse */
        static int ll=-1,mask=0;
        static int *buf=NULL;
        if (l!=ll) {
                int i;
                if (l>ll) buf=srealloc(buf,(1<<l)*sizeof(*buf));
                for(i=0;i<(1<<l);i++)
                        buf[i]=0;
                ll=l;
                mask=(1<<l)-1;
                }
        x&=mask;
        if (buf[x]==0 && x!=0) buf[x]=rev(x,l);
        return buf[x];
        }

int getnextbit(void) {
        /* returns the next bit (yes, bit) from stdin */
        static int n=0,v=0;

        if (v==EOF) return EOF; /* keep returning EOF when EOF is reached */
        if (n<=0) {
                v=getchar();
                if (v==EOF) return EOF;
                n=8; /* 8-bit chars */
                }
        n--;
        v<<=1;
        return !!(v&256);
        }

static int bitsource(void) {
        /* gets a bit from stdin or (memory) from otherstdin */
        if (otherstdin) {
                r=*(otherstdin++);
                if (r=='\0') {otherstdin--; r=EOF; return r;}
                }
        else {
                r=getchar();
                if (r==EOF) return r;
                }
        return r;
        }

static int getbit(void) {
        /* gets a bit from a text binary stream */
        static int r=0;
        if (r==EOF) return EOF;
        r=bitsource();
        if (r==EOF) return EOF;
        r&=1;
        return r;
```

```
        }

static int getnrzbit(void) {
        /* gets an NRZI bit from a text binary stream */
        static int r=0;
        int y;
        if (r==EOF) return EOF;
        y=r;
        r=bitsource();
        if (r==EOF) return EOF;
        r&=1;
        return (y!=r);
        }

int getnbits(int n) {
        /* return the next n bits from otherstdin or stdin, text binary */
        int r,x;
        r=getbit();
        if (r==EOF) return EOF;
        for(n--;n>0;n--) {
                x=getbit();
                if (x==EOF) x=0;
                r=(r<<1)|x;
                }
        return r;
        }

int getnnrzbits(int n) {
        /* return the next n bits from otherstdin/stdin, text binary NRZI */
        int r,x;
        r=getnrzbit();
        if (r==EOF) return EOF;
        for(n--;n>0;n--) {
                x=getnrzbit();
                if (x==EOF) x=0;
                r=(r<<1)|x;
                }
        return r;
        }

void outputnrz(int bit) {
        /* print an NRZI bit */
        static int lastbit=0;
        bit&=1;
        lastbit^=bit;
        putchar('0'+lastbit);
        }

void outputnrzbits(int n,unsigned int word) {
        /* print several NRZI bits */
        for(;n>0;n--,word>>=1) outputnrz(word);
        }

double fact(int n) {
        /* calculate factorials */
/* recursion free zone! */
static double *table=NULL;
static int tabsize=0,tabto=0;
int ts;

if (n<0) return 0.0;
ts=(tabsize<=0)?8:tabsize;
while(n>=ts) ts*=2;
if (ts>tabsize) {
        tabsize=ts;
        table=srealloc(table,tabsize*sizeof(*table));
        }
while(n>=tabto) {
        table[tabto]=(tabto>0)?table[tabto-1]*(double)tabto:1.0;
        tabto++;
```

```
            }
return table[n];
}
```

References

[1] William L. Abbott and John M. Cioffi. Timing recovery for adaptive decision feedback equalization of the magnetic storage channel. *IEEE Globecom 1990 (San Diego, CA, USA) Conference Record*, 3:1794–1799, November 1990.

[2] Kenneth Abend and Bruce D. Fritchman. Statistical detection for communication channels with intersymbol interference. *Proceedings of the IEEE*, 58(5):779–788, May 1970.

[3] AIM, 634 Alpha Drive, Pittsburgh, PA, 15238–2802, USA. *Guideline for measurement of effective magnetic parameters of magnetic stripes on media*, 1994.

[4] Ross Anderson and Markus Kuhn. Tamper resistance—a cautionary note. *The second.USENIX Workshop on Electronic Commerce Proceedings, Oakland, CA, USA*, pages 1–11, November 1996. ISBN 1–880446–83–9.

[5] Patrick Arnett, Ching Tsang, Tom Diola, and Lang Vo. TMR considerations at 5 Gb/in^2 and 10 MB/s. *IEEE Transactions on Magnetics*, 33(5):2674–2676, September 1997.

[6] Jonathan Ashley, Mario Blaum, Brian Marcus, and C. Michael Melas. Performance and error propagation of two DFE channels. *IEEE Transactions on Magnetics*, 33(5):2773–2775, September 1997.

[7] L.C. Barbosa. Simultaneous detection of readback signals from interfering magnetic recording tracks using array heads. *IEEE Transactions on Magnetics*, MAG-26(6):2163–2165, September 1990.

[8] Claude Berrou and Alain Glavieux. Near optimum error correcting coding and decoding: turbo-codes. *IEEE Transactions on Communications*, 44(10):1261–1271, October 1996.

[9] H. N. Bertram. *The Theory of Magnetic Recording*. Cambridge University Press, 1994.

[10] Kimberly C. Bracken, Hazim N. Zayed, and L. Richard Carley. Adaptive continuous-time equalization and FDTS/DF sequence detection. *IEEE Transactions on Magnetics*, 31(6):3048–3050, November 1995.

[11] Volker Braun. On impulse response measurement techniques for linear tape recording channels. *IEEE Globecom 1996 (London) Conference Record*, pages 963–967, November 1996.

[12] Micropædia Britanica, 1986. The Encyclopædia Britanica notes that Valdemar Poulson of Denmark made the first 'telegraphone' recordings on steel wire in 1900. Fifteenth edition, 7:682, 'magnetic recording'.

[13] British Standard. *BS-7106 Recording techniques for information on identification cards*, 1989. EN 27 811:1989, ISO 7811:1985.

[14] British Standard. *BS-7096 Guide to design and use of bank cards with a magnetic stripe that employs track 3*, 1990. EN 24 909:1989, ISO 4909:1987.

[15] King Ip Chan and Justin C-I Chuang. Required interleaving depth in Rayleigh fading channels. *IEEE Globecom 1996 (London) Conference Record*, pages 1417–1421, November 1996.

[16] Sheng Chen, Bernard Mulgrew, and Stephen McLaughlin. Adaptive Bayesian equalizer with decision feedback. *IEEE Transactions on Signal Processing*, 41(9):2918–2927, September 1993.

[17] Ta-Mu Chien. Upper bound on the efficiency of DC-constrained codes. *Bell System Techinical Journal*, 49:2267–2287, November 1970.

[18] Kiran Chopra, Lamar Nix, and Kevin D. Taberski. The effects of MR induced nonlinearities on channel performance. *IEEE Transactions on Magnetics*, 30(6):4200–4202, November 1994.

[19] J. Cioffi, W. Abbott, and K.D. Fisher. A survey of adaptive equalization for magnetic-disk storage channels. *Asilomar 22nd Conference on Signals, Systems and Computers*, 1:20–24, 1988.

[20] Maxoptix Corporation. Tahiti 1 product specification and OEM technical manual, April 1990. Document 1015225 (optical disk drive).

[21] D.H. Crawford, R.W. Stewart, and E. Toma. Digital signal processing for active noise control. *IEE Electronics & Communication Engineering Journal*, 9(2):81–89, April 1997.

[22] Stephen R. Cumpson, Barry K. Middleton, and Steven E. Stupp. A thick media recording model for quantitative study of media with arbitrary easy axis orientations. *IEEE Transactions on Magnetics*, 33(3):2405–2411, May 1997.

[23] Paul J. Davey. *Channel coding techniques for a multiple track digital magnetic recording system*. PhD thesis, University of Plymouth, 1994.

[24] P.J. Davey, T. Donnelly, and D.J. Mapps. Two dimensional coding for a multiple-track, maximum-likelihood digital magnetic storage system. *IEEE Transactions on Magnetics*, 30:4212–4214, 1994.

[25] W. Diffie and M.E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, IT-22(6):644–654, November 1976.

[26] T. Donnelly, D.J. Mapps, and R. Wilson. Real time monitoring of skew angle on a compact cassette multitrack magnetic tape system. *Journal of the Institution of Electronic and Radio Engineers*, 56(2):49–52, February 1986.

[27] Epson. *RX-80 operation manual*. Dot matrix printer reference manual.

[28] Lars Erup, Floyd M. Gardner, and Robert A. Harris. Interpolation in digital modems—part II: Implementation and performance. *IEEE Transactions on Communications*, 41(6):998–1008, June 1993.

[29] James Fife. New coding scheme boosts read channel performance. *Data Storage*, 4(8):45–52, September 1997. ISSN 1078–0920.

[30] Kevin D. Fisher, John M. Cioffi, William L. Abbott, Philip S. Bednarz, and C. Michael Melas. An adaptive RAM-DFE for storage channels. *IEEE Transactions on Communications*, 39(11):1559–1568, November 1990.

[31] G. David Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, March 1973.

[32] Peter K. Franaszek. Run-length limited variable-length coding with error propagation limitation. US patent no. 3,689,899, September 1972.

[33] Ayis Gallopoulos, Chris Heegard, and Paul H. Siegel. The power spectrum of run-length-limited codes. *IEEE Transactions on Communications*, 37(9):906–917, September 1989.

[34] Floyd M. Gardner. Interpolation in digital modems—part I: Fundamentals. *IEEE Transactions on Communications*, 41(3):501–507, March 1993.

[35] Monisha Ghosh and Charles L. Weber. Maximum-likelihood blind equalization. *Optical Engineering*, 31(6):1224–1228, June 1992.

[36] Grace L. Gorman, Lang Vo, Ben H.L. Hu, Ching Tsang, Jim Cser, and Jay Lindquist. Recording studies of sub-micron write heads by focussed ion beam trimming. *IEEE Transactions on Magnetics*, 33(5):2824–2826, September 1997.

[37] I.S. Grant and W.R. Phillips. *Electromagnetism*. John Wiley & Sons, second edition, 1990.

[38] The Guardian newspaper, 24 October 1996. 'Online', page 14.

[39] Ke Han and Richard Spencer. Comparison of different detection techniques for digital magnetic recording. *IEEE Transactions on Magnetics*, 31(2):1128–1133, March 1995.

[40] Simon Haykin. *Adaptive Filter Theory*. Prentice-Hall, 1986.

[41] Jay R. Hoinville and William G. Haines. Noise characteristics of keepered magnetic recording media. *IEEE Transactions on Magnetics*, 33(5):3034–3036, September 1997.

[42] N. Honda, A. Kikawada, K. Ouchi, and S. Iwasaki. Inter-track orthogonal coding for ultra high track density recording. *IEEE Transactions on Magnetics*, 31(6):3099–3101, November 1995.

[43] Ronald A. Iltis. A Bayesian maximum-likelihood sequence estimation algorithm for *a priori* unknown channels and symbol timing. *IEEE Journal on Selected Areas in Communications*, 10(3):579–588, April 1992.

[44] Ronald A. Iltis, John J. Shynk, and K. Giridhar. Bayesian algorithms for blind equalization using parallel adaptive filtering. *IEEE Transactions on Communications*, 42(2–4 pt. 2):1017–1032, February 1994.

[45] Ronald S. Indeck and Elias Glavinas. Fingerprinting magnetic media. *IEEE Transactions on Magnetics*, 29(6):4095–4097, November 1993.

[46] T.J. Jackson, D.J. Mapps, E.C. Ifeachor, and T. Donnelly. A compensation scheme for head-to-tape misregistration in multiple-track tape systems. *IEEE Transactions on Magnetics*, MAG-28(5):2904–2906, September 1992.

[47] George V. Jacoby. A new look-ahead code for increased data density. *IEEE Transactions on Magnetics*, MAG-13(5):1202–1204, September 1977.

[48] George V. Jacoby and Robert Kost. Binary two-thirds rate code with full look-ahead. *IEEE Transactions on Magnetics*, MAG-20(5):709–714, September 1984.

[49] Fred Jeffers. High-density magnetic recording heads. *Proceedings of the IEEE*, 74(11):1540–1556, November 1986.

[50] D.C. Jiles. *Introduction to Magnetism and Magnetic Materials*. Chapman & Hall, 1991.

[51] Finn Jorgensen. *The Complete Handbook of Magnetic Recording*. Tab Books, fourth edition, 1996.

[52] Aleksandar Kavčić and José M.F. Moura. Signal generation model for high density magnetic recording. *IEEE Globecom 1996 (London) Conference Record*, pages 973–978, November 1996.

[53] Aleksandar Kavčić and José M.F. Moura. Experimental validation of the triangle zig-zag transition model. *IEEE Transactions on Magnetics*, 33(5):2704–2706, September 1997.

[54] Zachary A. Keirn. An error margin analysis technique for fixed delay tree search and decision feedback detectors. *IEEE Globecom 1996 (London) Conference Record*, pages 958–962, November 1996.

[55] John G. Kenney and Roger Wood. Multi-level decision feedback equalization: an efficient realization of FDTS/DF. *IEEE Transactions on Magnetics*, 31(2):1115–1120, March 1995.

[56] M. Kenward. Copy-proof strip takes swipe at fraudsters. *Financial Times*, 7 October 1993.

[57] K. Komoda, T. Kira, R. Minakata, K. Nakai, T. Nakamura, and H. Deguchi. Combined thin-film heads for digital compact cassettes. *IEEE Transactions on Magnetics*, 29(6):3826–3828, November 1993.

[58] Victor Yu. Krachkovsky, Yuan Xing Lee, and Liu Bin. Error propagation evaluation for MDFE detection. *IEEE Transactions on Magnetics*, 33(5):2770–2772, September 1997.

[59] Patricia Lamiell. Smart card proponents take new tampering questions in stride. *Associated Press*, 26 September 1996.

[60] Peter T. Landsberg. *Thermodynamics and Statistical Mechanics*. Oxford University Press, 1978.

[61] Bin Liu and Yuan Xing Lee. The effect of MR head nonlinearity on MDFE and PRML performance. *IEEE Transactions on Magnetics*, 33(5):2764–2766, September 1997.

[62] Jian Lu, Dennis M. Healy, Jr., and John B. Weaver. Signal recovery and wavelet reproducing kernels. *IEEE Transactions on Signal Processing*, 42(7):1845–1848, July 1994.

[63] Nigel D. Mackintosh. A superposition-based analysis of pulse-slimming techniques for digital recording. *IERE Conference on Video and Data Recording*, pages 121–146, July 1979.

[64] D.J. Mapps, N. Fry, and D. Smith. Non-linear magnetoresistance. *Studies in Applied Electromagnetics and Mechannics*, pages 628–633, 1996. ISSN 1383–7281.

[65] D.J. Mapps and E. McQuade. Computer simulation of audio replay heads. *IERE Conference on Video and Data Recording*, July 1979.

[66] C. Denis Mee and Eric D. Daniel. *Magnetic Recording*, volume 3: Video, Audio and Instrumentation Recording. McGraw-Hill, Inc., 1988.

[67] Armin Miller. Recording and/or reproducing system. US patent no. 3,108,261, October 1963.

[68] Mondex International. Mondex issues pilot figures as payphones go live. *Mondex International Newsroom*, 19 October 1995. Information: Martin Jones at Band & Brown Communications.

[69] J.E. Monson, K.P. Ash, and R.E. Jones Jr. Self-heating effects in thin film heads. *IEEE Transactions on Magnetics*, MAG-20(5):845–847, September 1984.

[70] Jaekyun Moon and L. Richard Carley. Performance comparison of detection methods in magnetic recording. *IEEE Transactions on Magnetics*, 26(6):3155–3172, November 1990.

[71] K. Mueller and Muller M. Timing recovery in digital synchronous data receivers. *IEEE Transactions on Communications*, 24(5):516–531, May 1976.

[72] Sapthotharan K. Nair and Jaekyun Moon. Simplified nonlinear equalizers. *IEEE Transactions on Magnetics*, 31(6):3051–3053, November 1995.

[73] Kermit Norris. Channel capacity of charge-constrained run-length limited codes. *IEEE Transactions on Magnetics*, MAG-17(6):3452–3455, November 1981.

[74] Steven J. Nowlan and Geoffrey E. Hinton. A soft decision-directed LMS algorithm for blind equalization. *IEEE Transactions on Communications*, 41(2):275–279, February 1993.

[75] Dae Sun Oh, Won Gi Jeon, Yong Soo Cho, Hyung Woon Park, and Ki Ho Kim. Convergence analysis of a PLL for a digital recording channel with an adaptive partial response equalizer. *IEEE Globecom 1996 (London) Conference Record*, pages 979–983, November 1996.

[76] Dean Palmer, Pablo Ziperovich, Roger Wood, and Thomas D. Howell. Identification of nonlinear write effects using pseudorandom sequences. *IEEE Transactions on Magnetics*, MAG-23(5):2377–2379, September 1987.

[77] A.M. Patel. Zero-modulation encoding in magnetic recording. *IBM Journal of Research and Development*, 19:366–378, July 1975.

[78] Philips. DCC magnetic head, application note version 10.0. Type RP110R1/10 head, record and playback rotary reverse.

[79] Philips. *Digital compact cassette recorder 70DCC900 service manual.*

[80] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in* C. Cambridge University Press, second edition, 1992.

[81] Shahid U.H. Qureshi. Timing recovery for equalized partial-response systems. *IEEE Transactions on Communications*, 24:1326–1331, December 1976.

[82] Shahid U.H. Qureshi. Adaptive equalization. *Proceedings of the IEEE*, 73(9):1349–1387, September 1985.

[83] Mahadevan Ramesh, Richard H. Dee, and Kenneth H. Franzel. Dependence of Barkhausen noise on film parameters in shielded MR heads. *IEEE Transactions on Magnetics*, 29(6):3817–3819, November 1993.

[84] Gary C. Rauch, Jia Jay Liu, Sang Y. Lee, Istvan Boszormenyi, Chuan Gao, Jing Gui, David Kuo, Bruno Marchon, Stan Vierk, and Roger Malmhäll. Glass-ceramic substrates for 1 Gb/in^2 and beyond. *IEEE Transactions on Magnetics*, 32(5):3642–3647, September 1996.

[85] H. Shafiee, M. Melas, and P. Sutardja. Performance comparison of EPRML and peak detection in high density digital magnetic recording. *IEEE Transactions on Magnetics*, 29(6):4015–4017, November 1993.

[86] Hamid Shafiee. Timing recovery for sampling detectors in digital magnetic recording. *International Conference on Communications*, pages S18–5, 1996.

[87] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. University of Illinois Press, Urbana, Chicago, London, 1949.

[88] J. Skilling and R.K. Bryan. Maximum entropy image reconstruction: general algorithm. *Monthly notices of the Royal Astronomical Society*, 211:111–124, 1984.

[89] D. T. Tang and L. R. Bahl. Block codes for a class of constrained noiseless channels. *Information and Control*, 17:436–461, 1970.

[90] H.K. Thapar, N.P. Sands, W.L. Abbot, and J.M. Cioffi. Spectral shaping for peak detection equalization. *IEEE Transactions on Magnetics*, 26(5):2309–2311, September 1990.

[91] Adam F. Torabi and Steven B. Marshall. Effect of MR read non-linearities on extracted dipulse response. *IEEE Transactions on Magnetics*, 33(5):2716–2718, September 1997.

[92] Ching-Hsiang Tseng and Cheng-Bin Lin. A stop-and-go dual-mode algorithm for blind equalization. *IEEE Globecom 1996 (London) Conference Record*, pages 1427–1431, November 1996.

[93] Mathew P. Vea and José M.F. Moura. Magnetic recording channel model with intertrack interference. *IEEE Transactions on Magnetics*, 26(7):4834–4836, 1991.

[94] E.B. Vera. High density recording with partial response coding, Viterbi detection and run-length-limited coding. *Digital Processing of Signals in Communications, IERE*, pages 265–281, April 1981.

[95] Paul A. Voois and John M. Cioffi. Upper bounds on achievable storage density: a two-dimensional approach. *IEEE Transactions on Magnetics*, 33(1):844–854, January 1997.

[96] Tsutomu Wakabayashi and Yasuaki Serita. Magnetic recording in Advanced Photo System cameras. *IEEE Transactions on Magnetics*, 33(5):2659–2664, September 1997.

[97] Gregory K. Wallace. The JPEG still picture compression standard. *Communications of the ACM*, 34(4):30–44, April 1991.

[98] Bruce A. Wilson and Shan X. Wang. Linearizing the read process for write nonlinearity measurements. *IEEE Transactions on Magnetics*, 33(5):2692–2694, September 1997.

[99] R. Wood, D.A. Lindholm, and R.M. Haag. On the bandwidth of magnetic record/reproduce heads. *IEEE Transactions on Magnetics*, MAG-21(5):1566–1568, September 1985.

[100] Roger Wood. Enhanced decision feedback equalization. *IEEE Transactions on Magnetics*, 26(5):2178–2180, September 1990.

[101] Xinzhi Xing and H. Neal Bertram. Experimental studies of nonlinearities in MR heads. *Journal of Applied Physics*, 79(8):5883–5885, April 1996.

[102] T. Yamakoshi, Y. Shimano, and H. Yada. The effect of non-linear MR read-back distortion on a PRML channel. *IEEE Translations Journal on Magnetics in Japan*, 9(3):45–51, May 1994.

[103] Cheolwoo You and Hong Daesik. Blind adaptive equalization techniques using the complex multi-layer perceptron. *IEEE Globecom 1996 (London) Conference Record*, pages 1340–1344, November 1996.

[104] Ephiraim Zehavi and Jack K. Wolf. On runlength codes. *IEEE Transactions on Information Theory*, 34(1):45–54, January 1988.

[105] V. Zieren, M. de Jongh, A. Broese van Groenou, J.B.A. van Zon, P. Lasinski, and G.S.A.M. Theunissen. Ultrathin wear-resistant coatings for the tape bearing surface of thin-film magnetic heads for digital compact cassette. *IEEE Transactions on Magnetics*, 30(2):340–345, March 1994.

[106] V. Zieren, G. Somers, J. Ruigrok, M. de Jongh, A. van Straalen, W. Folkerts, E. Draaisma, F. Pronk, and T. Mitchell. Design and fabrication of thin film heads for the digital compact casette audio system. *IEEE Transactions on Magnetics*, 29(6):3064–3068, November 1993.

[107] Pablo A. Ziperovich. Performance degradation of PRML channels due to nonlinear distortions. *IEEE Transactions on Magnetics*, 27(6):4825–4827, November 1991.

Index