MASTER

Noise and intersymbol interference analysis of a high definition television transmission system

van Assche, P.J.F.

Award date:
1990

EINDHOVEN UNIVERSITY OF TECHNOLOGY
FACULTY OF ELECTRICAL ENGINEERING
TELECOMMUNICATIONS DIVISION EC

NOISE AND INTERSYMBOL INTERFERENCE ANALYSIS OF A HIGH DEFINITION TELEVISION TRANSMISSION SYSTEM.

by P.J.F van Assche

Report of the graduation work
accomplished from 04–09–1989 to 30–08–1990
Professor : prof. dr. ir. G. Brussaard
Supervisors : prof. dr. J.C. Arnbak
prof. dr. ir. J.B.H. Peek
ir. A.P. Verlijsdonk

Summary.

This report deals with transmission of a HD–MAC signal over a satellite channel. The HD–MAC system is to be designed with respect to both satellite and cable transmission. This implies that system elements have to serve specific needs of both transmission paths. However, optimising one transmission path does not automatically mean optimising the other transmission path. Compromises have to be made.

To investigate the properties of the HD–MAC transmission system noise and intersymbol interference (ISI) analyses have been carried out. In both analyses the system is considered as completely as possible; however, each analysis has been concentrated on the influence of one or more specific system elements. In the noise analysis the influence of Nyquist filtering and E7– or E8–emphasis filtering is investigated. The intersymbol interference analysis is concentrated upon the influence of incorrect operation of the sampler (a random– or static phase error) in combination with various kinds of Nyquist filtering. Both investigations are carried out using theoretical analysis and computer simulations.

It has been found that Nyquist filtering equally divided between transmitter and receiver provides the best performance in terms of signal–to–noise ratio at the output of the receiver. The E7 pre– and de–emphasis filtering is shown to give a maximum gain of 6 dB in signal–to–noise ratio at the output of the receiver. For E8–emphasis filtering the gain is at most 11 dB. The actually obtained gain is dependent on the characteristics of the transmitted signal, but will be approximately 3 dB for the E7 and 6 dB for E8 pre– and de–emphasis filtering. For the human eye noise at frequencies around 1.5 MHz has proven to be most disturbing. The advantages of using pre– and de–emphasis filtering are visual perceptible.

The sampling instant of the analog-to-digital converter is found to be very critical. Small deviations in the sampling instant give rise to a maximum amount of ISI of tens of percents. The sampling accuracy must therefore be within nanoseconds. For small deviations there is almost no difference in the maximum amount of ISI between nonideal Nyquist filters with a roll-off factor of 20 % and 10 %. Furthermore, nonideal Nyquist filtering shows a constant difference in ISI compared with ideal Nyquist filtering. When the sampler is afflicted with a static phase error the amplitude spectrum of a output signal is hardly affected, although high frequency components are more sensitive to the error than low frequency components.

Acknowledgement.

The author wishes to thank ir. Leon Vervoort of Philips Consumer Electronics Eindhoven (Video Processing Group). Not only for providing necessary and relevant information, but specially for spending time and showing enthusiasm in many discussions and for carefully reading and correcting the manuscript.

Table of Contents

1. Introduction.

Since the beginning of regular public broadcast about fifty years ago the evolution of television has been tremendous. In the beginning a television receiving set was an exclusive apparatus for the 'happy few' ; now about 600 million TV–sets are spread all over the world, offering an opportunity to watch a diverse variety of TV–programs. At this moment home video–recorders and video–disc sets are available, there is a widespread network of cable distribution and international broadcast is possible using TV–satellites.

New signal processing techniques, chip– and satellite technology seem to offer almost infinite perspective in this technical wonderland. But this is only so from a pure technical point of view, for social and economical considerations ask for a evolutionary development, rather than a revolutionary development. This restricts the application of new technical acquirements. In Europe the Eureka–95 project was started in 1986 to develop an evolutionary and compatible new television system. This resulted in the HD–MAC standard, where HD is the abbreviation of *High Definition* and MAC the abbreviation of *Multiplexed Analogue Components*. It denotes the most essential property of the system: The information parts – like color, luminance and sound – (the 'analogue components') are transmitted separately ('multiplexed') in time and recombined only in the receiver. The MAC–standard itself is developed in 1975 and accepted as the European standard for future television broadcast. The HD–MAC standard makes it possible to evolve in three or four steps towards a full High Definition Television (HDTV) system using the properties of MAC. HDTV is the common name for a television–system with high picture– and sound quality. The first TV–sets using some of the acquirements of the MAC–standard will be available in 1991/92. The introduction of the HD–MAC system is not to be expected before 1995.

The difference of HDTV with the existing TV-quality is expressed in (see also [1] and [2]):

* an increase in resolution in both horizontal and vertical direction by a factor two.
* wide-screen picture format. This implies not only an increase of the display surface, but also a change in the aspect-ratio. The aspect-ratio (the ratio of the screen-width and the screen-height, now 4:3) is increased to 16:9, to obtain a better connection to films and the human eye.
* possibilities for pay-television (so-called access-restriction).
* elimination of unwanted mutual influence of chrominance- and luminance information (so-called cross-color and cross-luminance).
* a high speed data channel for subtitling, teletext and other information services.
* possibility for multiple sound channels for broadcasting in various languages.
* hifi-stereo sound in Compact-Disc quality.

The MAC-standard transmits the TV-signal in 625 lines, with 2:1 interlacing and a 50 Hz field frequency. However, the HD-MAC camera produces a picture of 1250 lines and 1920 samples per line (due to the increase in resolution). This implies that special adaptations of the HD-MAC signal before transmission over the so-called MAC-channel have to be made. These adaptations include coding, bandwidth reduction and filtering.

Transmission of the more complex HD-MAC television signal over the defined MAC satellite channel requires special operations. Moreover, distribution of a HD-MAC signal in the existing cable network demands other adaptations: The most constraining factor here is that the bandwidth of the transmitted signal must be limited to 12 MHz, instead of the 27 MHz channel bandwidth that is available in satellite transmission. Compromises have to be made to ensure that both transmission

paths provide reliable communication. This, of course, has its effects on the elements used in the HD-MAC transmitter and -receiver. The most radical change is that Vestigal Sideband (VSB) modulation will be used in cable networks, instead of the Frequency Modulation (FM) which is used in the satellite transmission path.

This report deals with transmission of a HD-MAC signal over the MAC satellite-channel. The influence of noise and intersymbol interference is investigated in relationship to some of the elements of the HD-MAC system. These elements are chosen to be suitable for both satellite- and cable transmission, but only the satellite transmission path is investigated. The investigation is carried out using theoretical analysis and computer simulations.

A more detailed introduction into MAC, HDTV and HD-MAC can be found in [1] or [2].

2. Description of the HD-MAC satellite transmission chain.

The HD-MAC system for transmission of HDTV signals is a relatively new system, and its modulation parameters are not yet fully determined at the time of writing. In this chapter a brief description will therefore be given of the modulation parameters for the HD-MAC signal as used in this report. As we will focus our attention on the HD-MAC luminance signal, the description of the modulation parameters for chrominance- and sound/data signals is omitted. An introductory description of these parameters can be found in [1].

A schematic block diagram of the HD-MAC satellite transmission chain is shown in figures 4 and 5. In the following, a description of each device is given.

2.1. THE TRANSMITTER SIDE.

The HDTV camera.

The HDTV camera scans a picture in 1250 lines, taking 1920 samples per line, with a field rate of 50 Hz. The picture is 2:1 interlaced, resulting in a picture frequency of 25 Hz. With the aspect ratio of 16:9, the output is a digital data stream of approximately 60 million pixels/s.

The Bandwidth Reduction Encoder.

The Bandwidth Reduction Encoder (BRE) is a digital signal processor which manipulates the digital samples taken by the camera, so that they can be transmitted over a MAC compatible channel. In this report, the time compressing and the video multiplexing function of the TDM are considered to be in the BRE. This is but a conceptual change in the system, for only luminance signals will be examined. The operations of the BRE include therefore

picture coding, interlacing and time compressing of the video signal.

At the output of the BRE, the luminance signal is a digital 8–bit parallel data stream at a clock frequency of 20.25 MHz. These 8 bits correspond with a nominal voltage of 1 Volt peak–peak, where:

–0.5 Volt is a black luminance signal (digital '16').

0.5 Volt is a white luminance signal (digital '240').

The E7 nonlinear pre–emphasis network.

As a consequence of the greater baseband bandwidth, satellite transmission of HD–MAC leads to a greater noise sensitivity than for conventional television broadcasting, or even for D2–MAC equipment [3].

Among other techniques, E7 nonlinear pre–emphasis is used to diminish the effects of noise [4]. A block diagram of the E7 function is given in figure 1. Note that the dotted box (A) is the same as the de–emphasis function which is required in the receiver. The pre–emphasis function consists of the following processes.

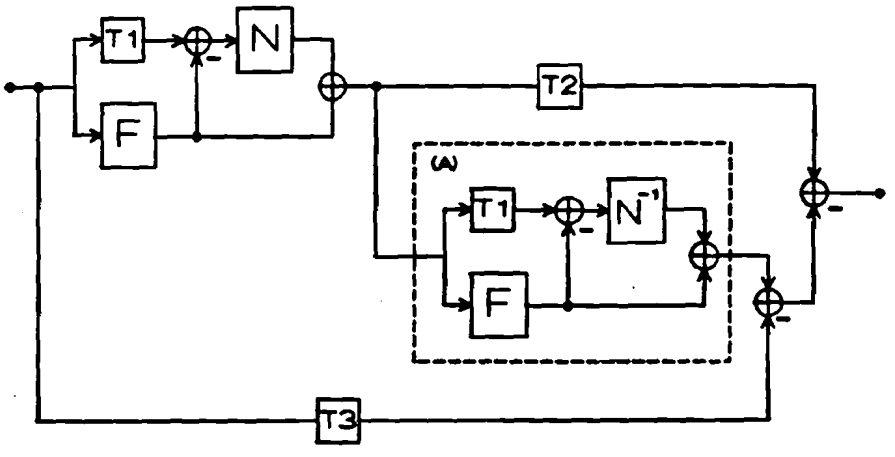

Figure 1: Block diagram of the E7 pre–emphasis network.

Low Pass Filter *F*.

The low pass filter *F* is a Gaussian filter, with a –3 dB bandwidth of 2 MHz. A 9 tap digital filter operating at a clock rate of 20.25 MHz is used with the following coefficients:

$$c_0 = 0.2986$$
$$c_1 = c_{-1} = 0.2255$$
$$c_2 = c_{-2} = 0.0972$$
$$c_3 = c_{-3} = 0.0251$$
$$c_4 = c_{-4} = 0.0029$$

Delay elements T_1, T_2, T_3.

The delay elements T_1, T_2, and T_3 are used to equalize the delay's through the different paths in the non–linear pre–emphasis network.

Non–linear function *N*.

The non–linear function *N* amplifies the low–amplitude parts of an incoming signal. *N* is described by the following equation (See figure 2):

$$V_o = \frac{V_i}{C} + \frac{1}{B} ln\left(\frac{V_i + \sqrt{V_i^2 + (2AC)^2}}{2AC}\right) \qquad (2.1)$$

where

$$A = 0.011$$
$$B = 19.80$$
$$C = 1.5225$$

The function N^{-1} in figure 1 is the inverse non–linearity to *N*. *N* and N^{-1} can be implemented by using a look–up table; this table has a (maximum) length of 256, since the input samples consist of 8–bit words.

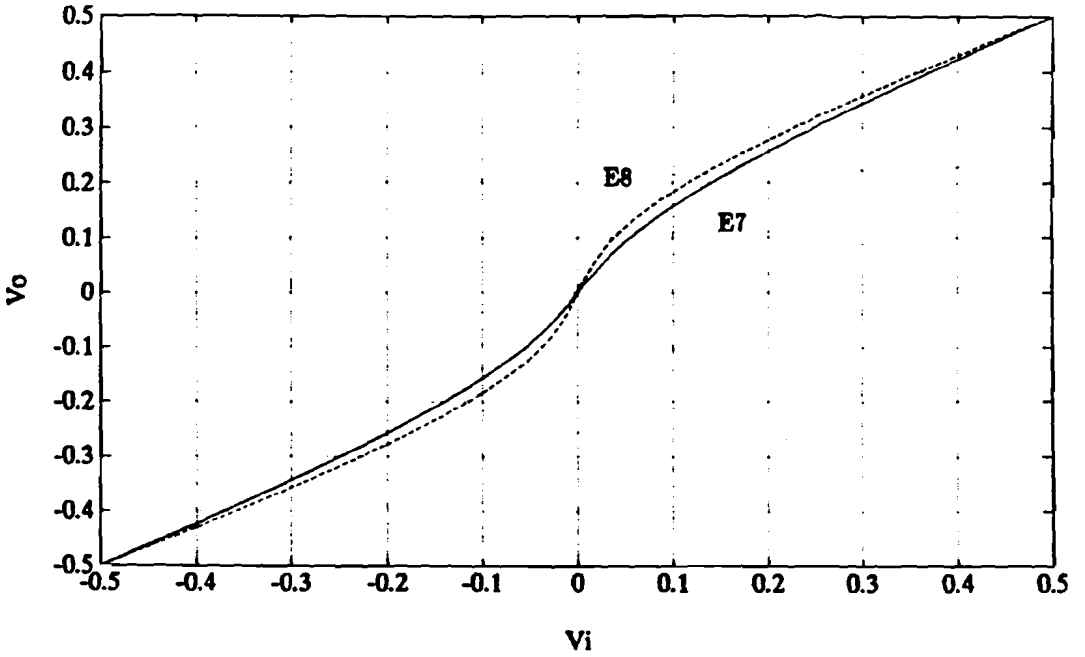

Figure 2: Non–linear function N of the E7 and the E8 pre–emphasis network.

Clearly, in the absence of the nonlinear functions, an overall flat frequency response is obtained. Further, it can be seen that the non–linear functions operate on the high frequency parts of the incoming signal only. Of the high-frequency parts, the low–amplitude components are most affected. For a low-frequency incoming signal the E7 function has no effect.

There has been a proposal for an alternative non–linear pre–emphasis filter. This filter, called E8, is the same as the E7 filter, but has a stronger non–linearity in its function N (See figure 2). The equation of its function N is also given by (2.1), but with the following coefficients:

$$A = 0.005$$

$$B = 19.80$$

$$C = 1.696$$

For compatibility reasons with respect to conventional MAC, the E8 filter shows some shortcomings. It is not decided yet which filter will be used in the HD-MAC system. We will use this filter in chapter 7, where noise power spectral densities are computed in a system with different de-emphasis filters.

The transmitter Nyquist filtering.

Theoretical analysis.

The coding of HD-MAC implies that the overall transmission channel must preserve the independence between the consecutive samples, at 20.25 MHz. This condition is fulfilled if the baseband channel meets the first Nyquist criterion, in the vicinity of 10.125 MHz [5,p.195]. In the HD-MAC system the Nyquist criterion is obtained by using a pulse shaping based on a raised cosine characteristic [5,p.195]. The necessary filtering for this shaping is equally shared between transmitter and receiver. The two resulting (equal) filters are called *half Nyquist* filters. The half Nyquist filter has a theoretical transfer function defined by the expression:

$$P_{\frac{1}{2}}(f) = \begin{cases} \sqrt{T} & |f| < \frac{1}{2T}(1-\alpha) \\ \sqrt{T}\cos\left\{\frac{\pi T}{2\alpha}\left(|f| - \frac{1}{2T}(1-\alpha)\right)\right\} & \frac{1}{2T}(1-\alpha) \le |f| \le \frac{1}{2T}(1+\alpha) \\ 0 & |f| > \frac{1}{2T}(1+\alpha) \end{cases} \qquad (2.2)$$

$\frac{1}{T} =$ 20.25 MHz.

$\alpha = 0.2$ or $\alpha = 0.1$;

where α is the so called roll-off factor ($0 \le \alpha \le 1$), a parameter indicating the 'smoothness' of the slope of the filter. α=0 corresponds to a (unrealizable) ideal low pass filter with cut off frequency $\frac{1}{2T}$, α=1 leads to a filter which uses the entire bandwidth up to $\frac{1}{T}$.

For the HD–MAC satellite transmission chain α=0.2 has proven to be an acceptable value (considering bandwidth, filter complexity and intersymbol interference sensitivity). A roll–off α=0.2 (also called 20% roll–off) results in a filter which occupies a bandwidth of $\frac{1+\alpha}{2T}$ = 12.15 MHz. However, account has to be taken to the use of existing cable networks for transmission of HD–MAC signals: For cable transmission a bandwidth of 12.15 MHz would cause adjacent channel interference problems, because the cable network only provides a 12 MHz channel spacing. Therefore, a compatible filtering solution is proposed, in which both satellite– and cable transmission use half Nyquist filters with a roll–off factor α=0.1, resulting in a bandwidth of 11.15 MHz. It is very likely that this proposal will be accepted.

Practical filtering.

Until thus far, we have considered Nyquist filtering as an analog filtering process, which, in most implementations, is the usual way of shaping digital pulses. In the HD–MAC transmission chain, however, pulse shaping is done by digital methods [6]. For this purpose, a transversal filter is used [5,p.228], with a digital input and a digital output.

Note that this requires some special signal manipulations: Since our clock frequency $\frac{1}{T}$ is 20.25 MHz, it is not possible (considering the Nyquist criterion) to design filters, or manipulate signals, which use frequencies higher than $\frac{1}{2T}$ = 10.125 MHz. But, since the half Nyquist filtering has a roll–off $\alpha \neq 0$, the digital filter has to operate in a frequency range above $\frac{1}{2T}$ = 10.125 MHz. This means that the half Nyquist filter sample frequency must be made higher than 20.25 MHz. In fact, the sample frequency is doubled to 40.5 MHz. The function of the digital half Nyquist filter is given by:

$$y(k\frac{T}{2}) = \sum_{j=0}^{N-1} c_j \, x((k-j)\frac{T}{2}) \qquad (2.3)$$

where $x(k\frac{T}{2})$ ($y(k\frac{T}{2})$) is a representation of the digital input (output) signal, N the number of taps of the filter, and c_j $(0 \leq j \leq N\text{-}1)$ the gains of the taps.

Clearly, the half Nyquist filter now expects an incoming digital data stream at a clock frequency of $\frac{2}{T} = 40.5$ MHz. Since we have an E7 pre-emphasis output at $\frac{1}{T} = 20.25$ MHz, the sampling frequency of this signal has to be upgraded to $\frac{2}{T} = 40.5$ MHz, *without* affecting its characteristics in the frequency domain. This is obtained by adding a zero between each two consecutive samples:

$$x'(k\frac{T}{2}) = \begin{cases} x(k\frac{T}{2}) & \text{if } k \text{ is even.} \\ 0 & \text{if } k \text{ is odd.} \end{cases}$$

Note that by doing so nothing changes in the frequency domain. (It is, in fact, identical to appending a trail of zeros to the DFT, the Discrete Fourier Transform).

Appendix A shows two realizable implementations of digital half Nyquist filters. It concerns a filter with 20% roll-off (α=0.2), using 15 taps, and a filter with 10% roll-off, using 31 taps (the 10% filter has a steeper slope and therefore requires more taps). Both filters are designed by use of the Remez algorithm.

To increase accuracy, the output of the half Nyquist filter in the HD-MAC system has a digital presentation in 12 bits, instead of 8 bits. Of course, the clock frequency remains 40.5 MHz throughout the rest of the transmitter.

The Time Division Multiplexer.

The Time Division Multiplexer (TDM) takes care of the packet structure of the HD-MAC baseband signal. Here, the sound/data signal is placed in the line- and field blanking intervals. The output of the TDM is the well-known line- and frame structure as described in detail in [7], [1] or [8]. While this research

report deals with luminance signals only, the TDM will be left out of consideration further on.

The Digital-to-Analog Converter.

The digital 12-bit luminance data are converted into real voltage levels in the Digital-to-Analog Converter (DAC). As seen before, the luminance signal measures 1 Volt peak-peak; with 12 bits input this means the DAC has an output voltage resolution of $1/4096 \approx 0.25$ mV. The output of the DAC is a stepwise (sample-and-hold) signal.

The Reconstruction Filter.

The Reconstruction Filter shapes the sample and hold signal from the output of the DAC into a time continuous signal. The stepwise DAC signal gives rise to a sinc fall-off in the frequency domain. Furthermore, the DAC signal uses a bandwidth far above than $\frac{1}{2T} = 10.125$ MHz. The reconstruction filter now deletes all frequency components above $\frac{1+\alpha}{2T}$ (baseband signal + roll-off) and amplifies the high frequency components to compensate the sinc fall-off. Analytically, the transfer function of the reconstruction filter is as follows:

$$H_{\mathrm{RF}}(f) = \begin{cases} \mathrm{sinc}^{-1}(f\frac{T}{2}) & 0 \leq |f| \leq \frac{1+\alpha}{2T} \\ 0 & f \text{ elsewhere.} \end{cases}$$

Note, that when the DAC and the reconstruction filter operate as the theory prescribes, the resulting function is that of an *ideal* low pass filter with cut off frequency $\frac{1+\alpha}{2T}$.

E1 linear pre-emphasis filtering.

To improve signal-to-noise ratio, E1 pre- and de-emphasis is applied. The pre-emphasis characteristic is defined by [8,p.63]:

$$H_{E1}(f) = A \cdot \frac{1 + j\,(f/f_1)}{1 + j\,(f/f_2)} \qquad (2.4)$$

where $A = \frac{1}{2}\sqrt{2}$

$f_1 = 0.84$ MHz

$f_2 = 1.5$ MHz.

Energy dispersal and DC restoration.

To avoid strong discrete components in the FM spectrum, which would cause interference to other users in the same frequency band, an energy dispersal signal is added to the whole baseband signal. In this report the energy dispersal signal plays no further role; for details the reader is referred to [8,p.63].

Just before the frequency modulation a carrier is added to the baseband signal in the DC restoration device. The DC restoration plays no role in our research, details can be found in [6].

Frequency modulation.

For satellite transmission the complete baseband HD-MAC signal is frequency modulated with a deviation of 13.5 MHz/V at the 0 dB crossover frequency of the E1 pre-emphasis network [8,p.63]. The channel bandwidth B is 27 MHz for a direct broadcast satellite (DBS) channel, as specified by the World Broadcasting-Satellite Administrative Radio Conference (WARC-BS) in 1977. With a baseband bandwidth f_x of $\frac{1+\alpha}{2T}$ Hz it follows that the deviation ratio

$$D = \frac{B}{2f_x} - 1 = \frac{4}{3(1+\alpha)} - 1$$

Since $D \ll 1$ the FM signal is called a narrowband FM (NBFM) signal.

The 12 GHz uplink.

In the 12 GHz uplink, the FM signal is converted (in two steps) up to the 12 GHz satellite channel. We will leave this device out of consideration.

2.2. THE RECEIVER SIDE.

On the receiver side (see figure 5), most functions are the exact reverse of the functions carried out in the transmitter. Where possible, reference will be made to the preceding text.

Receiver Front-end and IF-filter.

These devices take care of the down conversion of the 12 GHz satellite signal to the frequency modulated signal of bandwidth 27 MHz. Again, this is done in two steps: First down conversion to 1 GHz; second, down conversion to 70 MHz. Channel filtering and tuning takes place in the IF-filter.

The FM demodulator.

The FM signal is demodulated using a limiter discriminator or a phase locked loop (PLL). The demodulator sensitivity k_d equals $1/k_f$, so $k_d = \frac{1}{13.5}$ V/MHz.

$E1^{-1}$ linear de-emphasis filtering.

The $E1^{-1}$ function is the inverse of E1. Hence, we can write:

$$H_{E1^{-1}}(f) = \left(H_{E1}(f)\right)^{-1} = \frac{1}{A} \cdot \frac{1 + j\,(f/f_2)}{1 + j\,(f/f_1)} \qquad (2.5)$$

Low Pass Filtering at 14 MHz.

The Low Pass Filter (LPF) with a cut-off frequency of approximately 14 MHz rejects outband noise and the influence of adjacent channels. The cut-off frequency of 14 MHz is chosen to ensure a flat frequency response up to 12.15 MHz (in the case of 20 % Nyquist filtering).

Clamping and Automatic Gain Control.

In the clamping and Automatic Gain Control (AGC) circuit undesirable shifts

and amplifications in the voltage levels are corrected. This means that the original zero–voltage level is brought back to zero volts (by use of the so called 'clamp period' the TDM has inserted, see [7]) and, for the luminance signal, the 1 Volt peak–peak level is restored. The latter is done using a reference signal, or an averaging function.

The Analog–to–Digital Converter.

In the Analog–to–Digital Converter (ADC or sampler) the HDTV analog signal is brought back in its digital form. The sampling frequency is of course 40.5 MHz. In contradiction to the 8–bit luminance signal at the input of the DAC, the output of the ADC is a 9–bit digital signal, representing voltage levels between –1 and 1 Volt, thus taking care of possible overshoots. Overshoots can be caused by the Nyquist filtering, the E7 filtering or the MAC coding.

The Time Division Demultiplexer.

The Time Division Demultiplexer divides the different signals (luminance, chrominance, sound/data) to different outputs.

The Receiver Nyquist filtering.

The receiver half Nyquist filter is the same as the transmitter half Nyquist filter. Only the number of bits per incoming or outcoming sample is different: In the transmitter we had a digital input signal consisting of 8 bits per sample, and an output digital signal consisting of 12 bits per sample. In the receiver, the ADC delivers a 9–bit signal, so this is the input of the receiver half Nyquist filter. The output is, in conformity with the transmitter, a digital signal consisting of 8 bits per sample. Again, the filter operates at a sampling frequency of 40.5 MHz.

Subsampling.

Still, the sample frequency is $\frac{2}{T} = 40.5$ MHz, while this was only necessary to carry out the Nyquist filtering. At this point, Nyquist filtering has been done and we may return to the original sampling frequency of $\frac{1}{T} = 20.25$ MHz. This is carried out in the subsampling device: Of each two samples, the first is kept and the second thrown away. In the frequency domain, halving the sampling frequency means that the spectrum is 'folded'; as shown in figure 3a and 3b (see also, for example, [1]).

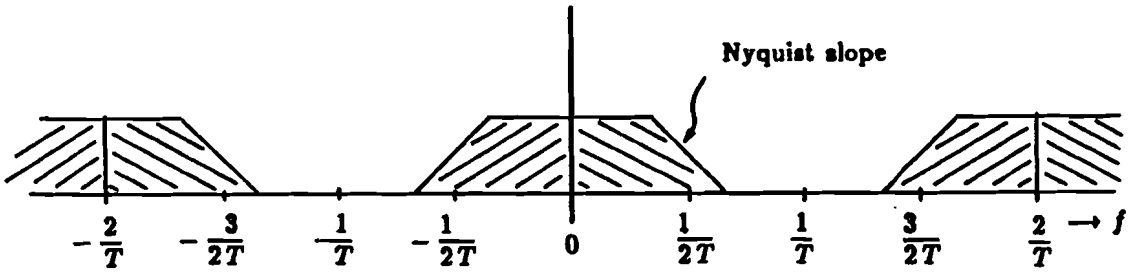

Figure 3a: Amplitude spectrum of the HD–MAC signal before subsampling: Sampling frequency is $\frac{2}{T} = 40.5$ MHs. Equal shaded denote equal spectrum contents.

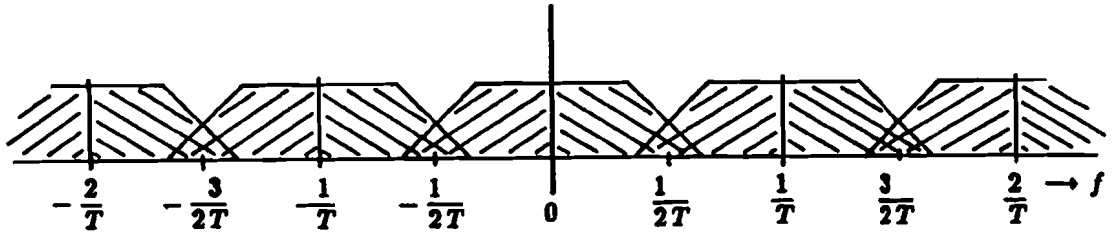

Figure 3b: Amplitude spectrum of the HD–MAC signal after subsampling: Sampling frequency is $\frac{1}{T} = 20.25$ MHs. Equal shaded denote equal spectrum contents.

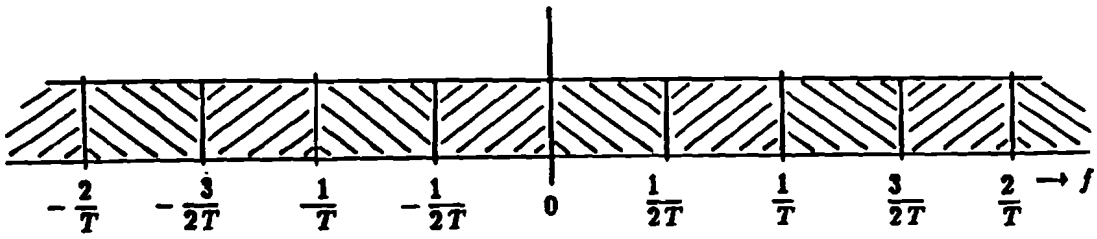

Figure 3c: Resulting amplitude spectrum of the HD–MAC signal after subsampling.

But, as (by definition of Nyquist filtering) the spectrum of figure 3a has a radially symmetric slope, the resulting spectrum becomes automatically the sampled at $\frac{1}{T} = 20.25$ MHz original baseband spectrum (with respect to E7 pre–emphasis filtering); see figure 3c.

The E7 non linear de–emphasis network.

As previously pointed out, the dotted box (A) in figure 1 is the E7 de–emphasis network as used in the receiver. The different parts of the de–emphasis network have also been described.

The Bandwidth Restoration Decoder.

The Bandwidth Restoration Decoder (BRD) performs the inverse operations of the BRE.

The HD–MAC system for satellite transmission of HDTV signals is now sufficiently specified for the research carried out in this report.

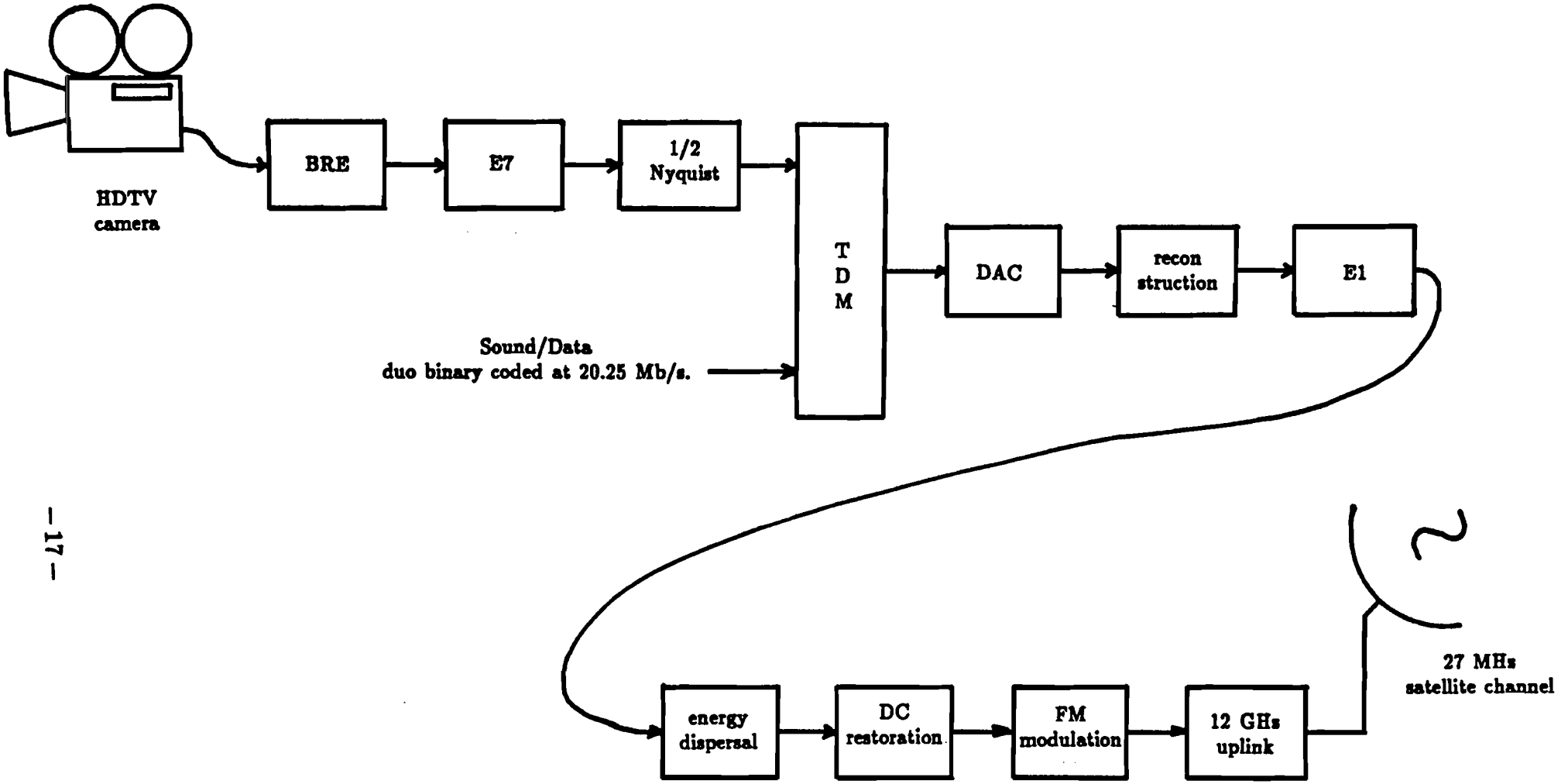

Figure 4: Block–diagram of the HDTV satellite transmission chain (transmitter–side).

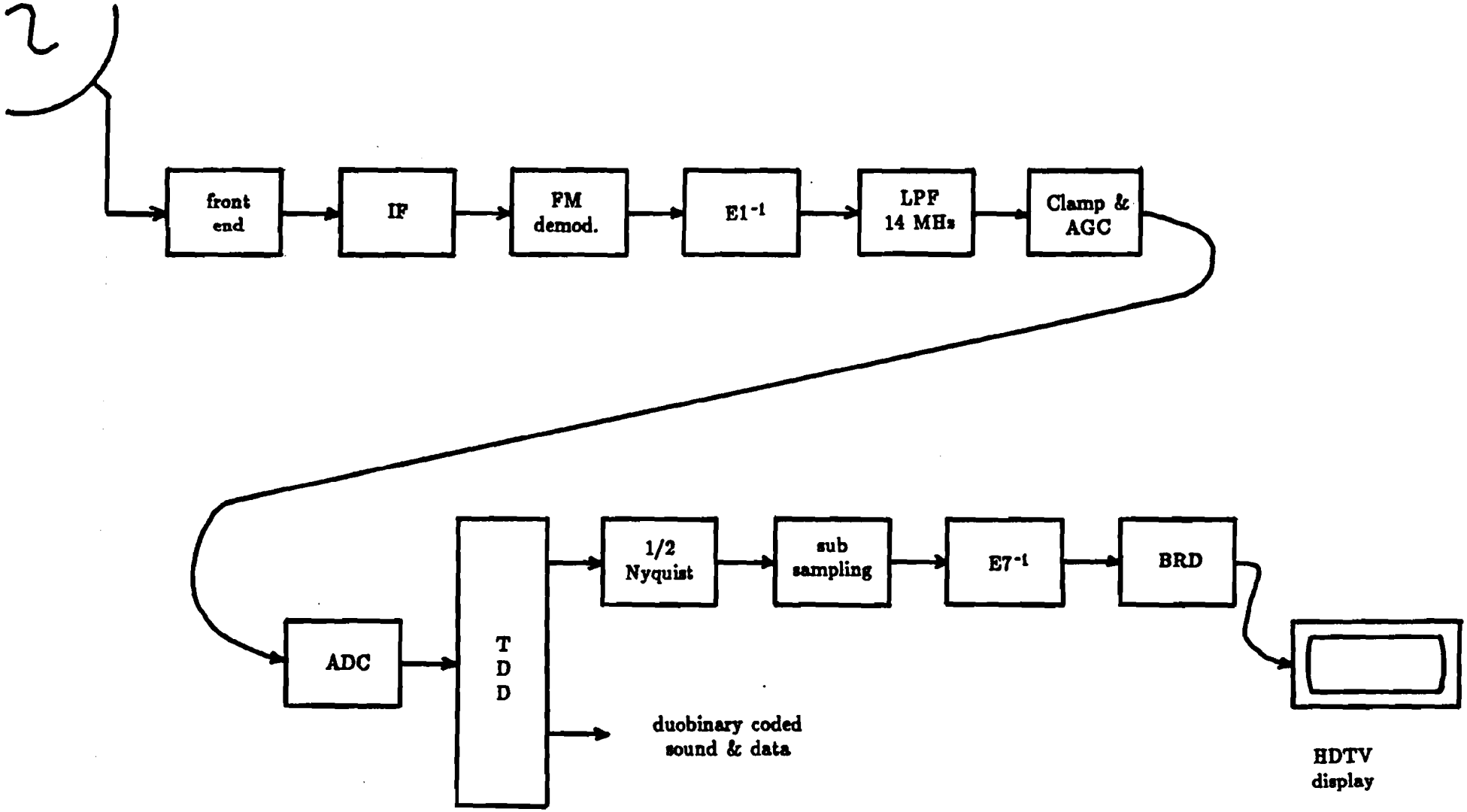

Figure 5: Block–diagram of the HDTV satellite transmission chain (receiver–side).

3. Differences in signal-to-noise ratio in a FM-system for three different Nyquist filter configurations.

As stated in chapter 2, and shown in figures 4 and 5, the Nyquist filtering of the HD-MAC signal is equally divided between the transmitter and receiver. This means that both the transmitter and the receiver contain a half Nyquist filter, as specified by equation (2.2). This is, of course, not the only solution possible, for one might consider transferring the whole Nyquist pulse shaping to the transmitter or the receiver. For a baseband data transmission system [5,chapter 5.2.2] has proved that the equal division of the Nyquist shaping gives the best performance in terms of signal-to-noise ratio at the output of the receiver. The question is: Is this result also valid for a system which uses frequency (de)modulation ?

To investigate this, we consider the simplified system given in figure 6. In this figure, H_T and H_R denote the transmitter and receiver filters respectively. $x(t)$ is the input data pulse train, S_T (S_R) the average signal power at the input (output) of the FM-system, and S_o the average signal power at the output of the receiver. n_i is additive Gaussian white noise (AGWN), causing a noise power N_o at the output of the receiver.

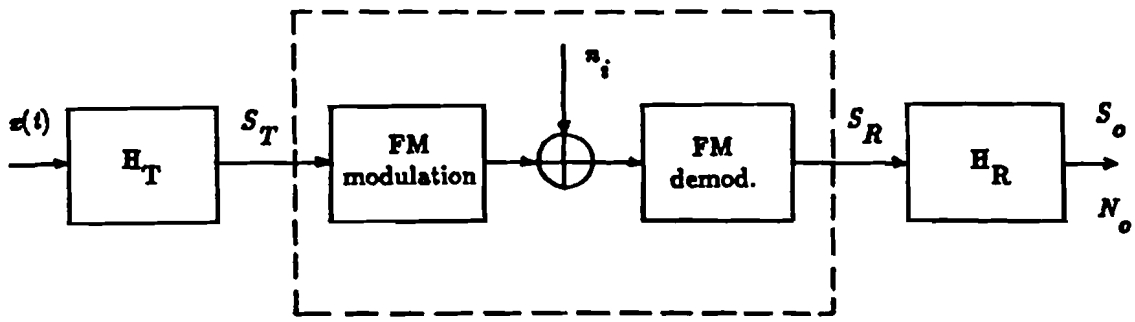

Figure 6: FM-system with transmitter and receiver filters.

By definition, the transmitter and receiver filter together must shape an incoming pulse into a raised-cosine characteristic. In case of perfect filtering this means:

$$X(f).H_R(f).H_T(f) = P_1(f) \tag{3.1}$$

where $X(f)$ is the Fourier transform of an incoming pulse, and $P_1(f)$ the raised cosine frequency characteristic:

$$P_1(f) = \begin{cases} 1 & |f| < \frac{1}{2T}(1-\alpha) \\ \cos^2\left\{ \frac{\pi T}{2\alpha}\left(|f| - \frac{1}{2T}(1-\alpha)\right)\right\} & \frac{1}{2T}(1-\alpha) \le |f| \le \frac{1}{2T}(1+\alpha) \\ 0 & |f| > \frac{1}{2T}(1+\alpha) \end{cases} \tag{3.2}$$

with α and T as defined in chapter 2. Note that, for computational comfort, the factor T is omitted in comparison with equation (2.2). Taking for $x(t)$ a train of delta pulses, the Fourier transform $X(f)$ of one pulse with amplitude one is $X(f) = 1$, so equation (3.1) reduces to:

$$H_R(f).H_T(f) = P_1(f) \tag{3.3}$$

With help of equation (3.3) we consider three different configurations for H_T and H_R:

1. Full Nyquist filtering in the transmitter:

$$H_T(f) = P_1(f)$$

$$H_R(f) = W(f) = \begin{cases} 1 & |f| \le \frac{1+\alpha}{2T} \\ 0 & f \text{ elsewhere.} \end{cases}$$

2. Half Nyquist in the transmitter, half Nyquist in the receiver:

$$H_T(f) = H_R(f) = \sqrt{P_1(f)}$$

3. Full Nyquist in the receiver:

$$H_T(f) = W(f)$$

$$H_R(f) = P_1(f)$$

In the following, for each configuration the output signal-to-noise ratio S_o/N_o is derived. S_T is taken as the reference signal power and therefore held constant in each computation.

3.1. FULL NYQUIST FILTERING IN THE TRANSMITTER (ad 1).

$$H_T(f) = P_1(f)$$
$$H_R(f) = W(f)$$

First, we derive the noise power at the output of the receiver. The noise power follows from its power spectral density (psd):

$$N_o = \int_{-\infty}^{\infty} G_{n_o}(f)\, df$$

where $G_{n_o}(f)$ is the noise psd at the output of the receiver. It is known [5,p.354] that the noise psd at the FM demodulator output is given by

$$G_n(f) = kf^2$$

with k a constant depending on channel and demodulator characteristics. Therefore, we can write (see also figure 7):

$$\begin{aligned} G_{n_o}(f) &= G_n(f).|H_R(f)|^2 \\ &= kf^2.|W(f)|^2 \end{aligned} \tag{3.4}$$

and

$$N_o = \int_{-\frac{1+\alpha}{2T}}^{\frac{1+\alpha}{2T}} kf^2\, df = \frac{k}{12}\left(\frac{1+\alpha}{T}\right)^3 \tag{3.5}$$

Second, the signal power S_o is derived. This is quite easy, since the transmitted energy S_T is fully set in band $|f| \leq \frac{1+\alpha}{2T}$, so the receiver filter $W(f)$ has no effect on the signal psd. Remember that in a frequency modulator-demodulator cascade, the output psd is the same as the input psd [5,p.353], so $S_R = S_T$, and

$$S_o = S_T \tag{3.6}$$

3.2. HALF NQYUIST IN THE TRANSMITTER, HALF NYQUIST IN THE RECEIVER (ad 2).

$$H_T(f) = H_R(f) = \sqrt{P_1(f)}$$
$$= \sqrt{T}\, P_{\frac{1}{2}}(f)$$

as given by equation (2.2). Again, with $G_{n_o}(f) = G_n(f).|H_R(f)|^2$ we can write:

$$G_{n_o}(f) = \begin{cases} kf^2 & |f| < \frac{1-\alpha}{2T} \\ kf^2 \cos^2\left\{ \frac{\pi T}{2\alpha}(|f| - \frac{1}{2T}(1-\alpha)) \right\} & \frac{1-\alpha}{2T} \leq |f| \leq \frac{1+\alpha}{2T} \\ 0 & |f| > \frac{1+\alpha}{2T} \end{cases} \qquad (3.7)$$

(see figure 7), and

$$N_o = \int_{-\infty}^{\infty} G_{n_o}(f)\, df$$
$$= 2\int_0^{\frac{1-\alpha}{2T}} kf^2\, df + 2\int_{\frac{1-\alpha}{2T}}^{\frac{1+\alpha}{2T}} kf^2 \cos^2\left\{ \frac{\pi T}{2\alpha}(f - \frac{1}{2T}(1-\alpha)) \right\} df \qquad (3.8)$$

while $G_{n_o}(f)$ is an even function around $f = 0$.

Using partial integration and a rainy sunday afternoon, the reader can verify that (3.8) reduces to:

$$N_o = \frac{k}{T^3}\left\{ \frac{(1-\alpha)^3}{24} + \frac{(1+\alpha)^3}{24} - 2\left(\frac{\alpha}{\pi}\right)^2 \right\} \qquad (3.9)$$

The calculation of S_o requires the psd of S_T, since $H_R(f)$ in this case does not give a flat frequency response. Let $G_{S_T}(f)$ be the psd of S_T, then we can write:

$$G_{S_T}(f) = X(f).|H_T(f)|^2$$
$$= |H_T(f)|^2 \qquad (3.10)$$

for $X(f) = 1$.

Furthermore,

$$S_T = \int_{-\infty}^{\infty} G_{S_T}(f)\,df = \int_{-\infty}^{\infty} |H_T(f)|^2\,df = \int_{-\infty}^{\infty} P_1(f)\,df = \frac{1}{T}$$

With equation (3.10) we have

$$\begin{aligned} G_{S_o}(f) &= G_{S_R}(f).|H_R(f)|^2 = G_{S_T}(f).|H_R(f)|^2 \\ &= |H_T(f)|^2.|H_R(f)|^2 = P_1^2(f) \end{aligned}$$

where $G_{S_R}(f)$ is the psd of S_R. Note that this equation is independent of the configuration of H_T and H_R.

The average signal power S_o at the output of the receiver is expressed by

$$\begin{aligned} S_o &= \int_{-\infty}^{\infty} G_{S_o}(f)\,df \\ &= \int_{-\infty}^{\infty} P_1^2(f)\,df \\ &= 2\int_0^{\frac{1-\alpha}{2T}} df \;+\; 2\int_{\frac{1-\alpha}{2T}}^{\frac{1+\alpha}{2T}} \cos^4\left\{\frac{\pi T}{2\alpha}\left(f-\frac{1}{2T}(1-\alpha)\right)\right\} df \\ &= \frac{1}{T}\left(1-\frac{\alpha}{4}\right) \end{aligned} \tag{3.11}$$

so

$$S_o = \left(1-\frac{\alpha}{4}\right) S_T \tag{3.12}$$

3.3. FULL NYQUIST IN THE RECEIVER (ad 3).

$$H_T(f) = W(f)$$
$$H_R(f) = P_1(f)$$

As in 3.1 and 3.2 we write (see figure 7):

$$G_{n_o}(f) = G_n(f).|H_R(f)|^2$$

$$= \begin{cases} kf^2 & |f| < \frac{1-\alpha}{2T} \\ kf^2 \cos^4\left\{ \frac{\pi T}{2\alpha}(|f| - \frac{1}{2T}(1-\alpha)) \right\} & \frac{1-\alpha}{2T} \leq |f| \leq \frac{1+\alpha}{2T} \\ 0 & |f| > \frac{1+\alpha}{2T} \end{cases} \qquad (3.13)$$

For the noise power N_o at the output of the receiver it then follows:

$$N_o = 2\int_0^{\frac{1-\alpha}{2T}} kf^2\,df + 2\int_{\frac{1-\alpha}{2T}}^{\frac{1+\alpha}{2T}} kf^2 \cos^4\left\{ \frac{\pi T}{2\alpha}(f - \frac{1}{2T}(1-\alpha)) \right\} df$$

with the solution

$$N_o = \frac{k}{T^3}\left\{ \frac{5(1-\alpha)^3}{96} + \frac{(1+\alpha)^3}{32} - 2\left(\frac{\alpha}{\pi}\right)^2 + \frac{\alpha}{8}\left(\frac{\alpha}{\pi}\right)^2 \right\} \qquad (3.14)$$

(This solution is obtained by use of $\cos^4(\phi) = \frac{3}{8} + \frac{1}{2}\cos(2\phi) + \frac{1}{8}\cos(4\phi)$ and partial integration).

Since $H_T(f) = W(f)$ we have

$$G_{S_T}(f) = |W(f)|^2$$

using equation (3.10), so

$$S_T = \int_{-\frac{1+\alpha}{2T}}^{\frac{1+\alpha}{2T}} df = \frac{1+\alpha}{T}$$

The signal power S_o is already given by equation (3.11). Using this expression for S_T, we have

$$S_o = \frac{1-\frac{\alpha}{4}}{1+\alpha} S_T \qquad (3.15)$$

3.4. DISCUSSION OF THE RESULTS.

Equations (3.4), (3.7) and (3.13) state the noise psd's for the three configurations of H_T and H_R. Figure 7 shows a drawing of these psd's for Nyquist filtering with 20% roll off.

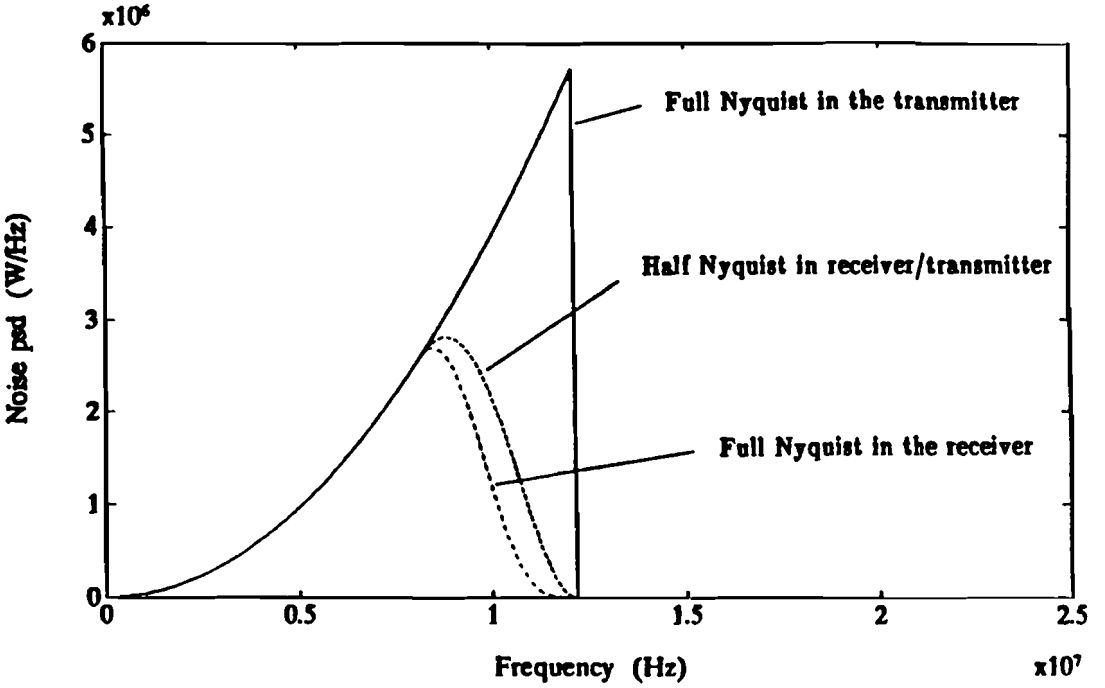

Figure 7: Noise power spectral densities for three configurations of H_T and H_R (α=0.2).

The curve for configuration 1 (full Nyquist in the transmitter) is the ordinary 'triangle noise' psd, as known in the FM theory [5,p.354]. It's sudden fall at 12.15 MHz is due to $W(f)$, the ideal low pass filter. The three curves are the same for $f <$ 8.1 MHz, for the Nyquist filtering affects higher frequencies only.

Equations (3.5), (3.9) and (3.14) express the noise power N_o for the three configurations, equations (3.6), (3.12) and (3.15) the signal power S_o.
They are resumed here:

1. Full Nyquist in the transmitter:

$$S_o = S_T$$

$$N_o = \frac{k}{12}\left(\frac{1+\alpha}{T}\right)^3$$

2. Half Nyquist in the transmitter, half Nyquist in the receiver:

$$S_o = (1-\frac{\alpha}{4})\, S_T$$

$$N_0 = \frac{k}{T^3}\left\{\frac{(1-\alpha)^3}{24} + \frac{(1+\alpha)^3}{24} - 2\left(\frac{\alpha}{\pi}\right)^2\right\}$$

3. Full Nyquist in the receiver:

$$S_0 = \frac{1-\frac{\alpha}{4}}{1+\alpha} S_T$$

$$N_0 = \frac{k}{T^3}\left\{\frac{5(1-\alpha)^3}{96} + \frac{(1+\alpha)^3}{32} - \frac{\alpha-16}{8}\left(\frac{\alpha}{\pi}\right)^2\right\}$$

With these equations, we have expressed the signal-to-noise ratio S_0/N_0 in terms of k, α, T and S_T. Of these terms, α, T and S_T are constant and k is depending on demodulator and channel characteristics. It is usual to express the signal-to-noise ratio at the output of the receiver as a function of the signal-to-noise ratio at the input. For this, we introduce the term C/N; the signal-to-noise ratio at the input of the FM demodulator. The question now arising is: How can k be expressed in terms of C/N?

By definition, k is given by [5,p354]:

$$k = \frac{k_d^2}{2C}(2\pi)^2\eta$$

with k_d the demodulator sensitivity and $\eta/2$ the psd of the additive white Gaussian channel noise. For the noise power N we have

$$N = \int_{27\ \text{MHz}} \frac{\eta}{2}\,df = \eta\ 13.5\ .10^6$$

because the channel bandwidth is 27 MHz. Putting k and N together, we have

$$k = \frac{k_d^2}{27.10^6}(N/C)$$

S_0/N_0 as function of C/N is shown in figure 8 for α=0.2 and S_T = 0.5 W. S_T = 0.5 W is chosen, while the voltage levels at this point lie roughly between -0.5 and 0.5 Volt (see chapter 2). $k_d = \frac{1}{13.5}$ V/MHz, in accordance with chapter 2.

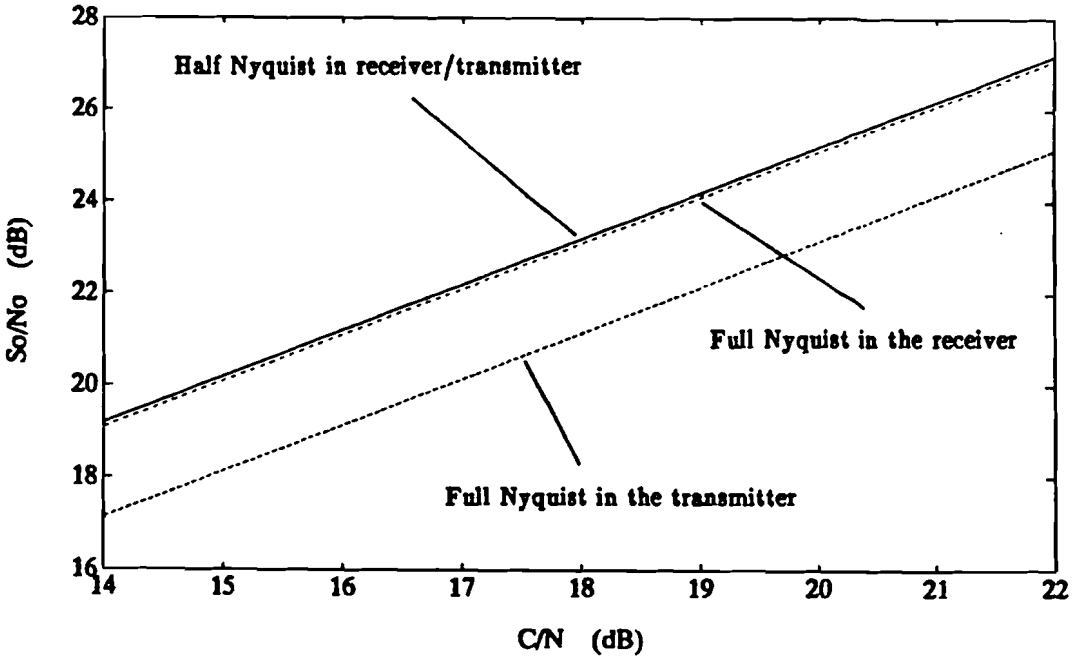

Figure 8: Output signal–to–noise ratio as function of the input signal–to–noise ratio for α=0.2 .

3.5. CONCLUSIONS.

From figure 8 it can be seen that the differences in S_0/N_0 for the FM–system are the same as for the baseband system. The best configuration of the three has Nyquist filtering equally divided between transmitter and receiver. A very close suboptimum is putting the Nyquist filter in the receiver. Full Nyquist filtering in the transmitter gives a significant degradation in performance.

As full Nyquist in the transmitter would require a sharp low pass filter in the receiver, no hardware would be saved by doing so. On the other hand, a full Nyquist filter in the receiver leads to a more complex filter than when needed for a half Nyquist filter. Therefore, no hardware can be saved by altering the filter configuration.

4. Description of the software.

In order to test a part of the HD–MAC transmission chain, most of the devices described in chapter 2 are implemented in software. With this computer simulations can be carried out to analyse the performance of the system in the presence of noise and in case of missampling.
The first part of this chapter describes which parts of the HD–MAC system are implemented and what assumptions have been made. In the second part an example is given to show the properties of the software.

4.1. NOTES ON THE IMPLEMENTATION.

The computer simulation deals with almost the complete path of the luminance signal through the system given by figures 4 and 5. For various reasons different devices of figures 4 and 5 have not been implemented.

Complexity : The BRE and BRD are too complex to implement on a personal computer. This fact limits the possibilities of the software, while the output of the BRD is the best measure of what is really to be seen on the HDTV display. Since noise or sampling errors are only interesting when causing a visible degradation, it might affect the impact of the results. To overcome this problem, a simple 'worst case' BRD model will be used where necessary (See chapter 6).

Matter of interest : Since we look at luminance signals, the time division (de)multiplexer is of no use. Furthermore, the energy dispersal signal and the DC restoration, as well as the clamping and AGC circuit will be left out of consideration. This is possible while only additive white Gaussian noise is investigated and no adjacent channel interference is assumed.

Neglect of elements : Using 'triangle noise' (see chapter 3.4) at the output of the FM-modulator, we assume the input noise to be white and Gaussian. Then, the 12 GHz uplink, front-end receiver and the intermediate frequency (IF) filter have no influence on the signal nor on the noise.

Altogether, the system of figure 9 is used for analysis. In the analysis, the following elements are considered ideal: the DAC, the reconstruction filter, the E1 pre– and de–emphasis, the FM (de–)modulation and the low pass filter at 14 MHz. These are all the 'analog' elements in the system. The rest of the elements are supposed to be according to their description in chapter 2; for the Nyquist filtering transversal filters are used with coefficients as specified in appendix A.

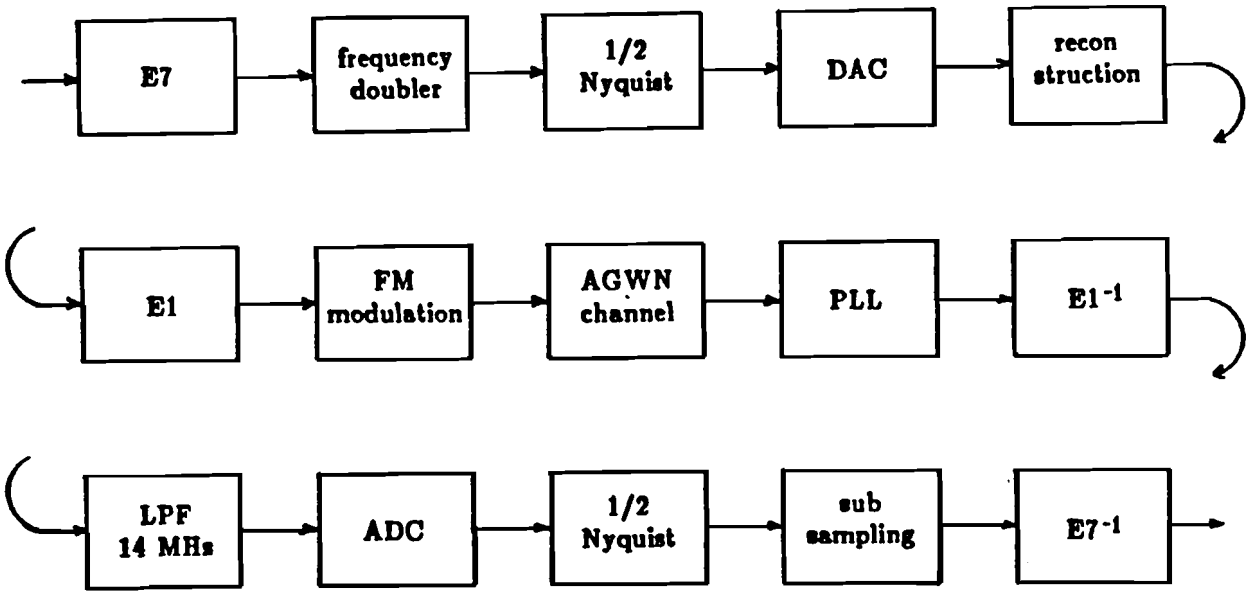

Figure 9: Part of the HD–MAC transmission chain as used for software implementation.

The implementation is written in so called Matlab–files; files which are supported by Matlab, a powerful software–packet for fast vector arithmetic. For detailed information on Matlab we refer to the Matlab reference manual [9]. A listing of the

programs can be found in appendix B.

One exception has been made for the FM–demodulator. For this device a Phase Locked Loop (PLL) is implemented in Pascal, instead of using Matlab–files. This is done because of speed considerations: The files written in Matlab are very slow in handling loops (although the possibility exists), while the Pascal language can deal with loops very efficiently. The Pascal implementation of the PLL can be found in appendix C. For the communication between the Pascal and Matlab files, the reader is also referred to appendices B and C.

For the E7 pre– and de–emphasis filters, the use of a non–linear function N and N^{-1} is necessary. For this, two look–up tables are loaded. The look–up tables are generated by a Pascal program originally written by L. Vervoort of Philips Consumer Electronics Eindhoven. It has been modified to serve the specific needs of the E7 pre/de–emphasis implementation. A listing of the Pascal implementations can also be found in appendix C.

The simulations of the system given in figure 9 has been made such, that long data files can be simulated in parts. Therefore, the initial and final conditions for the delays in each device are taken into account. Details can be found in the program listings in appendix B. Appendix D contains a 'mother'–file: A Matlab–file which simulates the HD–MAC transmission chain by calling the functions given in appendix B. The mother file also shows the use of piecewise simulation of long data files.

4.2. AN EXAMPLE: PROCESSING OF A STEP FUNCTION.

With the software, the response of the HD–MAC system on input of a step function can be obtained. This is done in the absence of channel noise and with 10% Nyquist filtering. The consecutive outputs of the elements of the system are given in figures

10.a–m. In these figures digital signals are represented by their corresponding voltage levels in 'staircases' (figures 10.a–d and 10.i–m). Remember, however, that they are 8,9 or 12 bit *digital* signals, according to the specifications given in chapter 2.

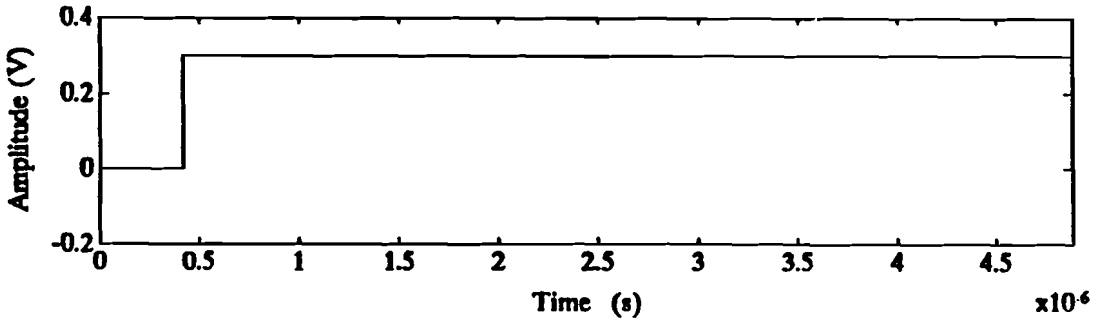

Figure 10.a: Input signal of the E7 pre–emphasis filter.

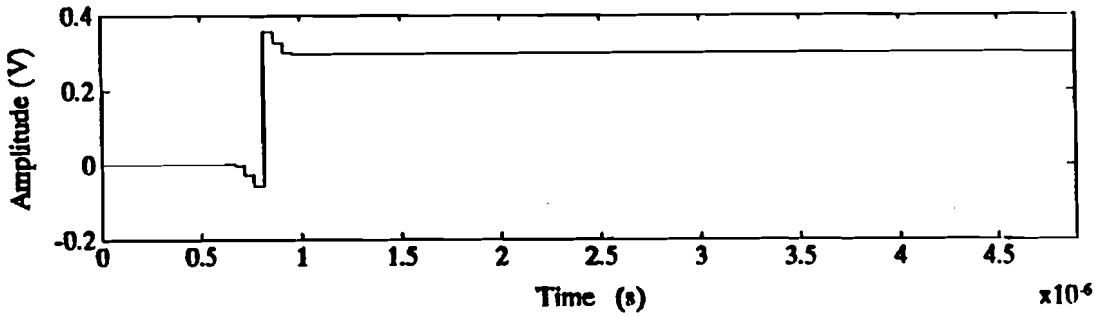

Figure 10.b: Output signal of the E7 de–emphasis filter.

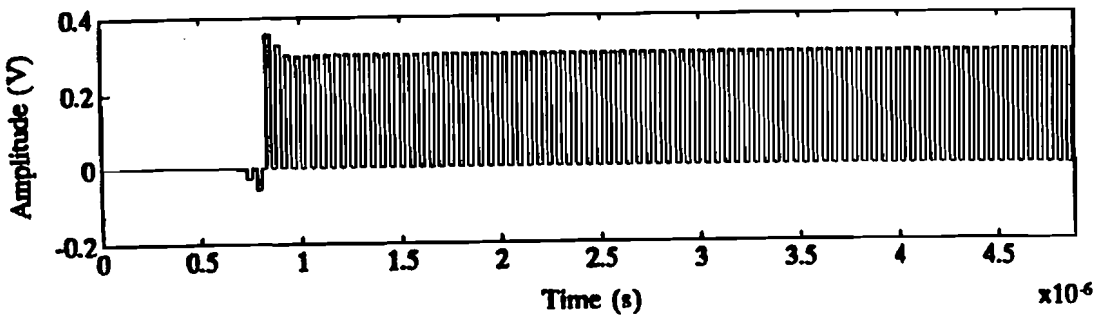

Figure 10.c: Output signal of the frequency doubler.

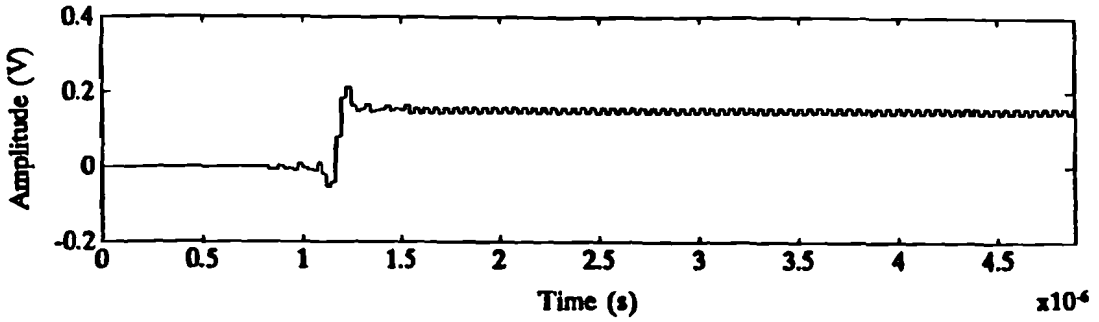

Figure 10.d: Output signal of the first 1/2 Nyquist filter.

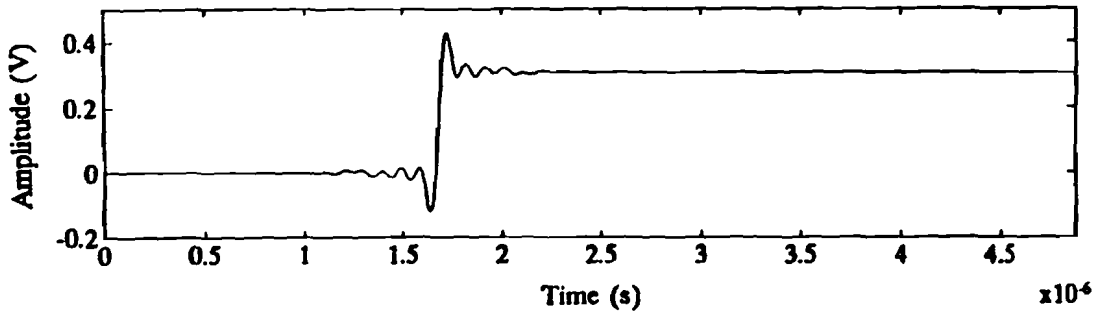

Figure 10.e: Output signal of the Reconstruction filter.

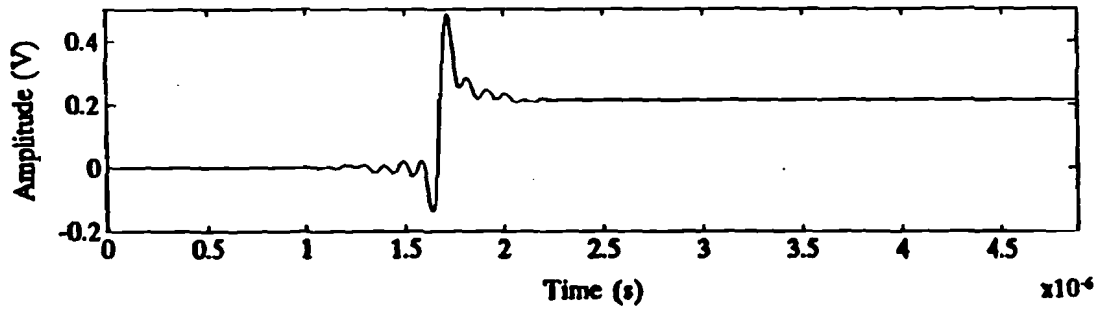

Figure 10.f: Output signal of the E1 pre-emphasis filter.

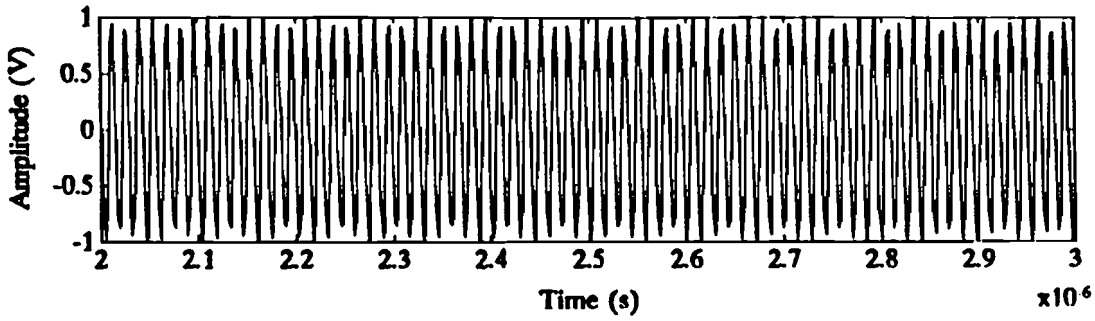

Figure 10.g: Output of the FM–modulator.

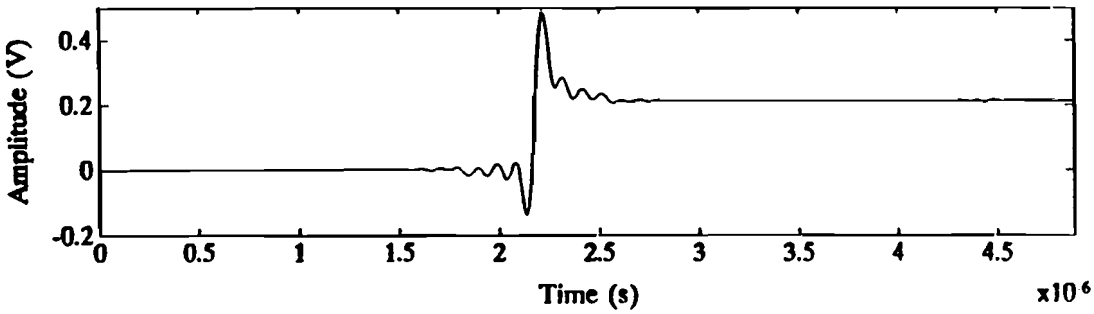

Figure 10.h: Output of the PLL.

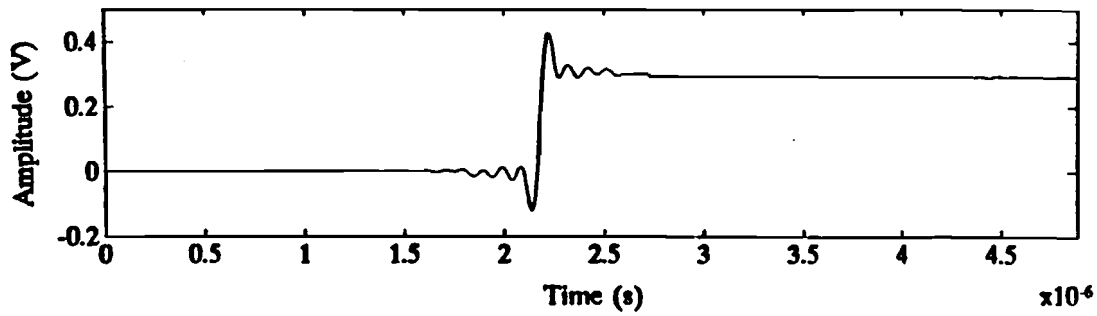

Figure 10.i: Output of the LPF at 14 MHz.

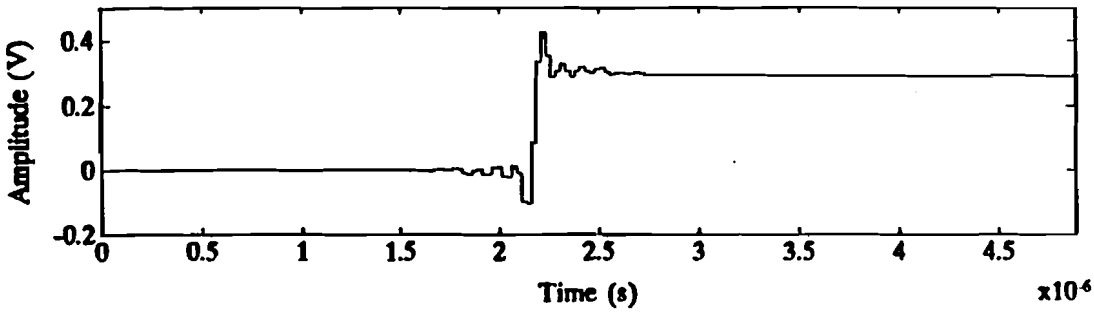

Figure 10.j: Output of the ADC.

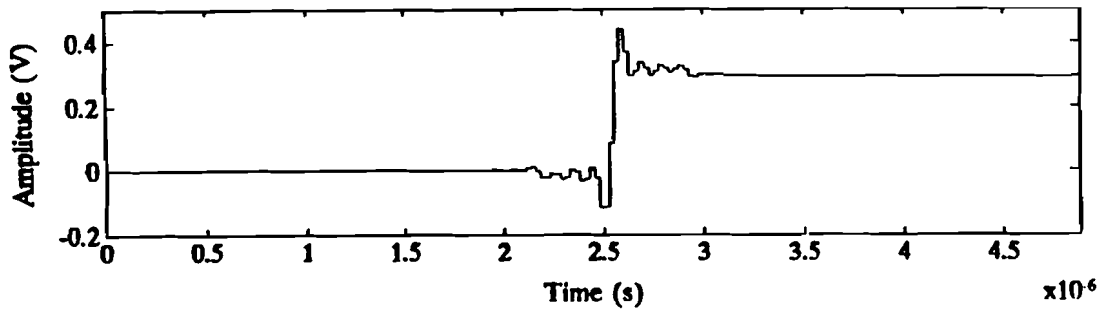

Figure 10.k: Output of the second 1/2 Nyquist filter.

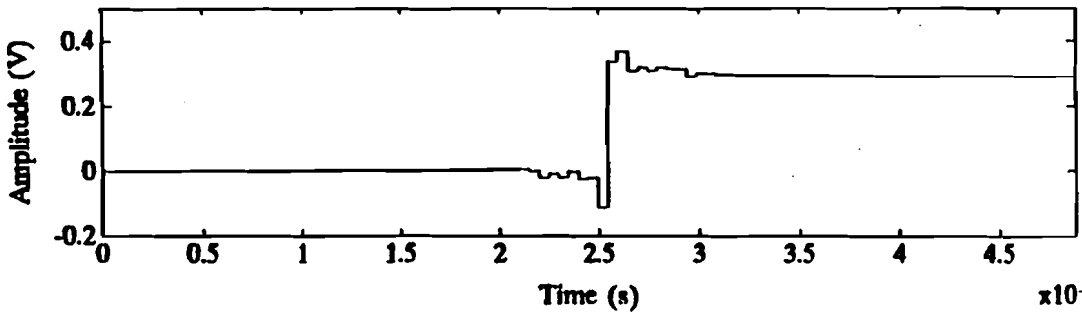

Figure 10.l: Output of the subsampler.

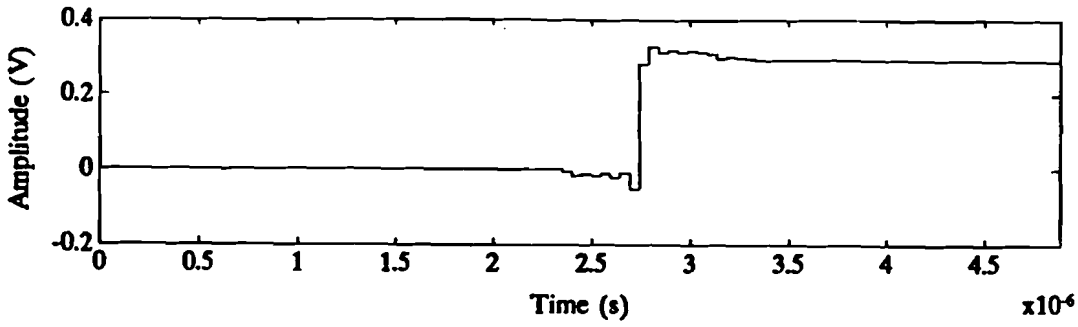

Figure 10.m: Output of the E7 de-emphasis filter.

A remarkable conclusion which can be derived from figure 10.m is that, even in the absence of noise, the outcoming signal is *not* error-free. The distortion originates from imperfections of the Nyquist and (in a smaller measure) of the E7 filtering. These distortions can hardly be seen on the HDTV display [3], so they are acceptable; trying to remove them would require more complex (and more expensive !) Nyquist filters. However, one of the consequences of this conclusion is that it is of no use trying to determine a output bit error rate in noise conditions, for there will always be bit errors, depending on the input signal. Other ways have to be found to measure the performance of the system in noise and jitter circumstances.

5. Effects of a random phase error in the sampling instant.

In the described system, Nyquist filtering is used to design an overall pulse shaping that would yield minimal distortions of the outcoming waveforms. However, in practical systems distortions will inevitably occur due to imperfect filter realisation, due to noise and changes in channel characteristics. The pulse shaping is made such that the sampler is as little critical as possible in these circumstances. In communication engineering this is often visualised by displaying what is known as the *eye pattern* [5,p.237]. The sensitivity of the system to timing errors is proportional to the shape and the width of the eye opening. Variations in the sampling instant (mostly called *jitter*) will cause influence of one pulse on another, for the overall shaping of the transmitted signal is the sum of all the individually Nyquist shaped pulses. For example, a sampling error for a higher level pulse surrounded by two higher level pulses will therefore be less critical than for a higher level pulse surrounded by two lower level pulses. The residual effect of adjacent pulses on a pulse is called the *intersymbol interference* (ISI). In this report, an undesirable variation in the sampling instant is called a *phase error* (as distinct from amplitude errors, where undesirable level variations occur).

In this chapter the influence of a random phase error is investigated. With the adjective random we limit ourselves in this report that the sampling instant is varying according to a Gaussian probability distribution with zero mean and deviation σ. The influence of the random phase error is dependent on σ , the incoming signal and the kind of Nyquist filtering that is used (leaving channel noise out of consideration). Its effect is an amount of ISI: The amplitude of the outcoming pulse will be changed. Chapter 5.1 derives the maximum amount of ISI that can be expected when using ideal or nonideal Nyquist filters, in chapter 5.2 we consider the distribution function of the ISI when dealing with a random phase error.

5.1. THE MAXIMUM AMOUNT OF ISI AS CONSEQUENCE OF A RANDOM PHASE ERROR.

Applying the foregoing remarks on the HD–MAC system, where two digital half Nyquist filters are used, we consider the system as shown in figure 11:

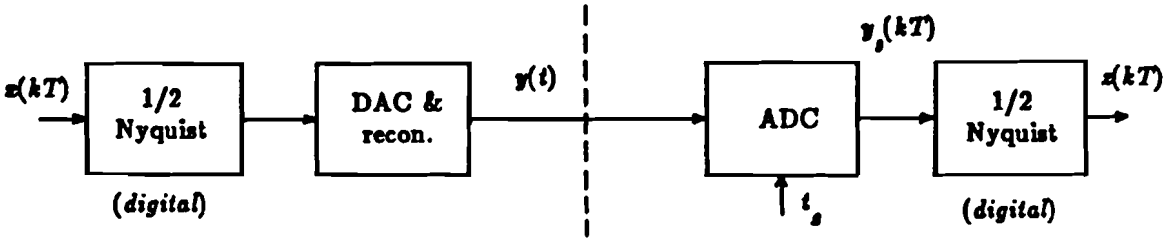

Figure 11: Pulse shaping parts of the HD–MAC system.

In figure 11, the dashed line denotes the separation between transmitter and receiver. $x(kT)$ is the input signal, $y(t)$ the output of the transmitter and input to the receiver. $y_s(kT)$ denotes the sampled version of $y(t)$, where the sampling instants are taken from t_s , a clock input. Finally, the output of the receiver gives us $z(kT)$, the processed signal $x(kT)$. When using ideal Nyquist filtering, ideal DAC and reconstruction and correct sampling, $z(kT)$ equals $x(kT)$. As pointed out in chapter 2.1, the ideal DAC and reconstruction filter together form an ideal low pass filter with cut–off frequency $\frac{1+\alpha}{2T}$. Since we are interested in 'mistuning' of the sampler in relationship to imperfections of the Nyquist filters, the DAC and the reconstruction filter are considered to be ideal.

Now, the system is explained and the problem is clear. But; how is ISI to be measured? Consider, for example, transmission of a zero pulse (corresponding to a grey luminance point) surrounded by pulses of higher absolute amplitudes (more black

or white luminance points): The output $z(kT)$ at the zero pulse time will be a zero, drowned in ISI. Only receiving ISI, so what is the conclusion?

To derive a meaningful measure we look at the theory of binary data transmission. In this theory only ± 1 is transmitted. Hence, a measure for ISI can be determined by transmission of one pulse and looking at the output signal values aside the wanted pulse time. All these 'remnants' will cause ISI on other pulses. For the system of figure 11 we can use a similar method:

1. Take for $x(kT)$ a pulse of arbitrary amplitude:
$$x(kT) = \begin{cases} x_0 & k = 0 \\ 0 & k \neq 0 \end{cases}$$
2. Process $x(kT)$ according to figure 11 and obtain $z(kT)$.
3. A measure for the maximum amount of ISI is now given by the sum of all the response values outside $z(0)$. Normalizing on $z(0)$ we then have:
$$\mathrm{ISI}_{\max} = \frac{\sum_{k \neq 0} |z(kT)|}{|z(0)|} \cdot 100\ \% \qquad (5.1)$$

The amplitude x_0 can be chosen arbitrary, while the system is linear so $z(kT) = x_0 . z_1(kT)$ and x_0 is always divided out in equation (5.1).
Equation (5.1) describes what percentage of the voltage of a transmitted pulse will contribute to ISI on other pulses. Putting it the other way around: Sending a train of pulses of amplitude $\pm x_0$, equation (5.1) gives the maximum amount of ISI that can be expected at any sampling instant. For example, with $\mathrm{ISI}_{\max} = 50\ \%$ it might be possible to receive $z(kT) = \frac{1}{2}x(kT)$ instead of $z(kT) = x(kT)$.

5.1.1. Ideal Nyquist filtering.

In case of ideal Nyquist filtering it is possible to go somewhat further into equation (5.1). Let $p_{\frac{1}{2}}(t)$ be the time domain impulse response of an ideal half Nyquist filter, as

specified in equation (2.2). Then, the taps of the ideal digital half Nyquist filter are given by $p_{\frac{1}{2}}(kT)$, where the idealisation lies in the fact that k runs from $-\infty$ to ∞ and no time delay is assumed. However, cascading the ideal digital half Nyquist filter and the ideal low pass filter results in the response of the ideal (analog) half Nyquist filter $p_{\frac{1}{2}}(t)$. We now have an easy description of the transfer functions in figure 11.

Taking $x_0 = 1$ and following the described procedure, we have

$$y(t) = p_{\frac{1}{2}}(t)$$

In correct operation the sampler takes its sampling instants at

$$t_s = iT \qquad (i \in \mathbb{I})$$

A random phase error is introduced by adding a random variable ΔT_i to t_s . ΔT_i is taken from a standard Gaussian probability distribution with mean zero, so $\Delta T \sim N(\mu=0,\sigma)$ and

$$t_s = iT + \Delta T_i$$

Hence

$$y_s(iT) = p_{\frac{1}{2}}(iT+\Delta T_i)$$

The receiver digital half Nyquist filter is also ideal, with transfer function $p_{\frac{1}{2}}(iT)$, so for output $z(kT)$ we have

$$z(kT) = (p_{\frac{1}{2}} * y_s)(kT)$$

where * denotes the discrete convolution operation. Writing out results in

$$\begin{aligned} z(kT) &= \sum_{i=-\infty}^{\infty} p_{\frac{1}{2}}(iT)\, y_s(kT-iT) \\ &= \sum_{i=-\infty}^{\infty} p_{\frac{1}{2}}(iT)\, p_{\frac{1}{2}}(kT-iT+\Delta T_k) \\ &= (p_{\frac{1}{2}} * p_{\frac{1}{2}})(kT+\Delta T_k) \end{aligned}$$

Since $p_1 = p_{\frac{1}{2}} * p_{\frac{1}{2}}$ is by definition the transfer function of an ideal *full* Nyquist filter, we have

$$x(kT) = p_1(kT+\Delta T_k)$$

Equation (5.1) can therefore be written as

$$ISI_{max} = \frac{\sum_{k\neq 0} |p_1(kT+\Delta T_k)|}{|p_1(\Delta T_k)|} \cdot 100\ \% \qquad (5.2)$$

$p_1(t)$ is the time domain impulse response of the well known raised cosine frequency characteristic (see chapter 2.1 or [5,p.195]), given by

$$p_1(t) = \frac{\cos(\pi\alpha\frac{t}{T})}{1-(2\alpha\frac{t}{T})^2} \cdot \mathrm{sinc}(\frac{t}{T}) \qquad (5.3)$$

Unfortunately, the random character of ΔT prevents us from calculating equation (5.2). However, using equation (5.2), simple simulations can be carried out. A program listing for such a simulation can be found in appendix D. In this program a sequence ΔT_k with $\Delta T \sim N(\mu=0,\sigma)$ is generated; then for each ΔT_k equation (5.2) is evaluated to obtain $ISI_{max}(k)$. The resulting ISI_{max} is the mean value of the sequence $ISI_{max}(k)$. This process is repeated for different values of σ . Figure 12 shows the results of the simulation for ideal Nyquist filtering with a roll–off factor of 10 % and 20 %.

From figure 12 it can be seen that the sampling instant is very critical, both for 20 % and 10% Nyquist filtering. But, as expected, 10 % Nyquist filtering is more critical than 20 %. A deviation ratio of $\sigma = 5$ ns (= $T/10$!) already gives us a maximum amount of ISI of 30 % respectively 40 %. This is critical, while $\sigma = 5$ ns means that approximately 30 % of the sample instants have a phase error of more than 5 ns (on account of the definition of deviation in a Gaussian probability distribution). Even for ideal Nyquist filtering we can therefore conclude that small random deviations in the sampling instant give rise to maximum ISI distortions of tens of percents.

5.1.2. Nonideal Nyquist filtering.

Equation (5.2) can also be derived for the system of figure 11 using the nonideal half Nyquist filters as specified in appendix A. However, this would result in a quite complex expression for $p_1(t)$ (in analogy with equation (5.3)), showing no advantage of insight. Rather than losing ourselves in mathematical worries, it is easier to use equation (5.1) for determination of ISI_{max}. The simulation for equation (5.1) differs from the one for equation (5.2) in that we now have to simulate the whole system shown in figure 11 instead of one expression. But, as can be seen in the listing of the simulation program in appendix D, there is only little increase in complexity. The results of the simulation for nonideal Nyquist filtering can be found in figure 12. Again, a 20 % and a 10 % roll-off factor is used.

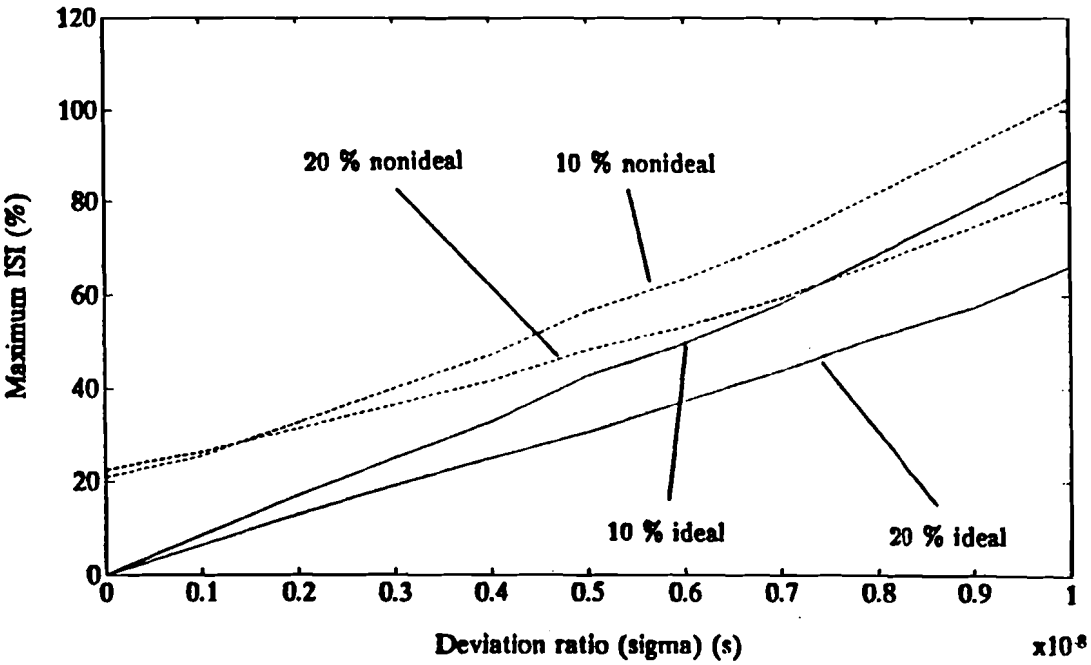

Figure 12: Maximum ISI due to a random phase error for ideal- and nonideal Nyquist filtering.

It immediately strikes that even for correct sampling ($\sigma = 0$) a maximum ISI

distortion of 20 % can occur. This, of course, is due to the fact that the filters are imperfect realisations. Furthermore, the nonideal filters show a similar behaviour as the ideal filters; the difference is that for each deviation ratio the nonlinear filters produce an approximately 20 % higher maximum amount of ISI.
For deviation ratio's smaller than 2 ns there is almost no difference between the 10 % and the 20 % filter in the maximum amount of ISI.

5.2. STATISTICAL PROPERTIES OF A SAMPLER WITH A RANDOM PHASE ERROR.

In chapter 5.1 we have seen that an imperfect sampler in combination with Nyquist filtering can produce large amounts of ISI. In this chapter the behaviour of the isolated sampler is investigated. If the sampling instant varies according to a Gaussian distribution function, the question is what the statistical characteristics of the outcoming samples of an imperfect sampler will be. Looking at figure 11, we are only concerned with the signals $y(t)$, t_s and $y_s(kT)$.

Figure 13 visualizes the functioning of a sampler in operation. Signal $y(t)$ is sampled with a sampler whose sampling instants show a timing shift ΔT_k, resulting in errors in $y_s(kT)$. $y_s(kT)$ will be a somewhat 'noisy' signal; its statistical characteristics can be easily approximated as follows.

Let ξ_k be the error caused by the imperfect sampling of $y(t)$, so (see figure 13)

$$\xi_k = y(kT+\Delta T_k) - y(kT)$$

Assuming ΔT_k to be positive (for negative ΔT_k the argumentation is similar) and $y(t)$ to be linear in the interval $(kT,kT+T)$ we have

$$y(t) \approx y(kT) + (t-kT).\frac{y(kT+T) - y(kT)}{T}$$

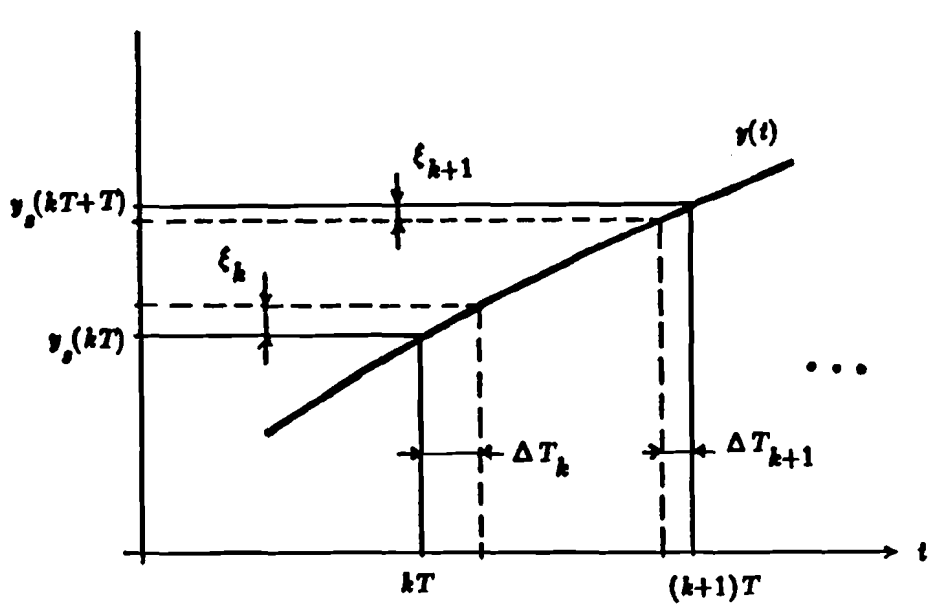

Figure 13: Sampling of $y(t)$ with a jittered sampler.

$$\approx y(kT) + (t-kT).r_k \qquad (kT \leq t \leq kT+T)$$

where $r_k = \dfrac{y(kT+T) - y(kT)}{T}$.

Hence,

$$y(kT+\Delta T_k) \approx y(kT) + \Delta T_k.r_k$$

and

$$\xi_k \approx \Delta T_k.r_k$$

Since r_k is constant and ΔT_k a random Gaussian variable with deviation σ , ξ_k is also a random Gaussian variable with deviation $\sigma_{\xi_k} = r_k.\sigma$.

The incorrect sampler can now be decomposed in terms of a correct sampler and an addition of Gaussian white noise, shown in figure 14.

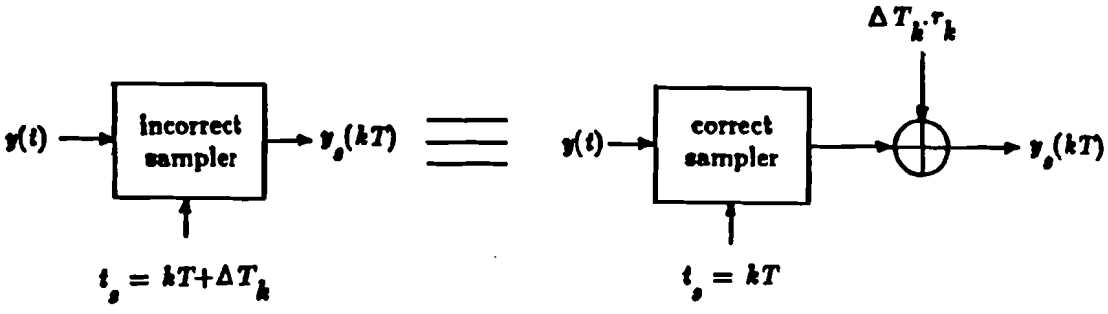

Figure 14: Equivalent block diagram of a jittered sampler.

6. Effects of a static phase error in the sampling instant.

In chapter 5 we investigated the influence of a random phase error in the sampling instant. Another malfunctioning of the sampler occurs when its sampling instants are exact $\frac{1}{2T}$ seconds from each other, but always a certain time away from the wanted sampling instant (" The rhythm is good, but it's out of tune "). In this case the sampler has a *static phase error*. The static phase error is expressed in seconds; for example, if the sampler has a static phase error of $\Delta T = 5$ ns, it means that the sampling instants are always 5 ns late.

In this chapter three effects of the static phase error are considered. First, the maximum amount of ISI is derived, as in chapter 5.1. Second, the influence of a static phase error on input of a block pulse train is considered. The last section is concerned with changes in the frequency spectrum as a consequence of a static phase error.

6.1. THE MAXIMUM AMOUNT OF ISI AS CONSEQUENCE OF A STATIC PHASE ERROR.

Most of the work for this problem has already been carried out in chapter 5.1. Figure 11 shows again the pulse shaping parts of the HD-MAC system to be considered and the expression for ISI_{max} in equation (5.1) is independent of the distribution of the phase error and hence still valid.

6.1.1. Ideal Nyquist filtering.

A similar argumentation as in 5.1.1 can be obtained for ideal Nyquist filtering in combination with a sampler which has a static phase error. The only difference is that the random noise variable ΔT_k is not random anymore, but a fixed (static) value,

$$\Delta T_k = \Delta T \qquad \text{for every } k \in \mathbb{I}$$

Equation (5.2) can therefore be rewritten as

$$ISI_{max} = \frac{\sum_{k \neq 0} |p_1(kT + \Delta T)|}{|p_1(\Delta T)|} \cdot 100\ \% \qquad (6.1)$$

with $p_1(t)$ as given in equation (5.3).

Equation (6.1) is a simple expression that can be easily analysed; no simulations are necessary when ΔT is constant. Figure 15 shows a plot of equation (6.1) for roll-off factors $\alpha = 0.2$ and $\alpha = 0.1$.

The curves in figure 15 for ideal Nyquist filtering are (approximately) straight lines through the origin. It can be seen that a static phase error can cause a large amount of ISI. For $\Delta T = 5$ ns ($= T/10$!) a maximum amount of ISI of 40 % respectively 50 % is possible. The sampling is therefore very critical and requires accuracy within nanoseconds.

6.1.2. Nonideal Nyquist filtering.

Also for nonideal Nyquist filtering the complexity of the analysis decreases. As in chapter 5.1.2 equation (5.1) for determination of ISI_{max} is simulated; but, since ΔT is constant, no long sequence ΔT_k has to be generated. The listing of a simulation program for determination of ISI_{max} for nonideal Nyquist filtering can be found in appendix D. Figure 15 shows the results of this simulation.

From figure 15 it can be seen that a static phase error in combination with nonideal Nyquist filtering shows a similar behaviour as for the ideal Nyquist filtering. The imperfections of the used Nyquist filters cause an increase in the maximum amount of ISI of approximately 15 % and 10 %, for 20 % respectively 10 % Nyquist

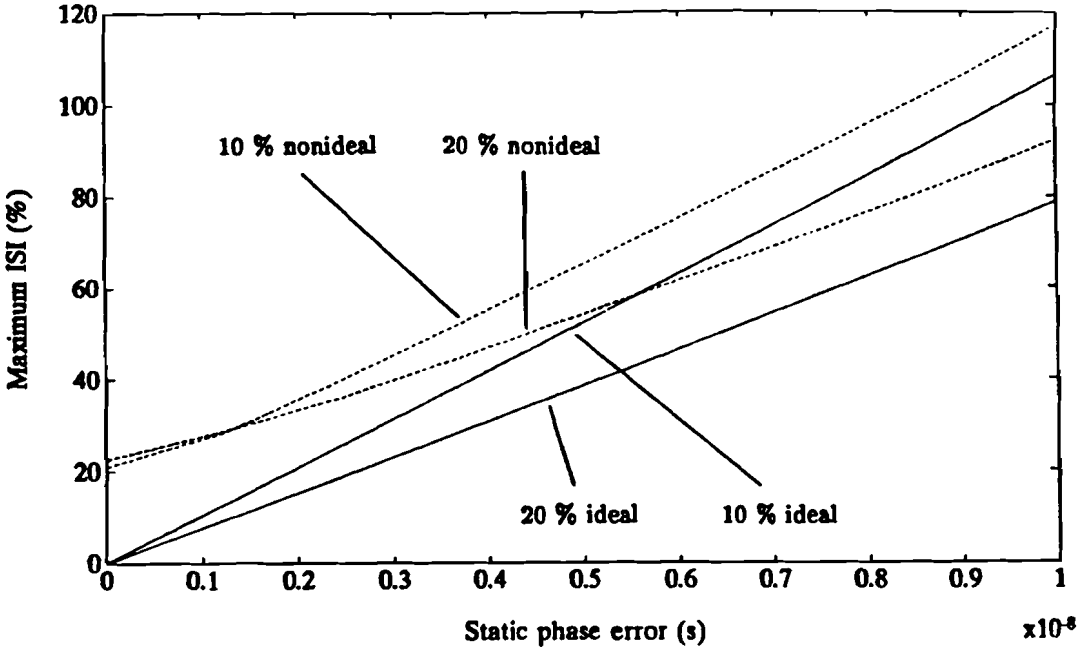

Figure 15: Maximum ISI due to a static phase error for ideal– and nonideal Nyquist filtering.

filtering. The results for this simulation are approximately equal to the results obtained in the simulation with the random phase error. For static phase errors smaller than 2 ns there is practically no difference between the two nonideal Nyquist filters.

Resuming, we can conclude that both from the point of view of random phase errors and from the point of view of static phase errors the sampling instant must be quite accurate (within nanoseconds).

6.2. A CASE STUDY: EFFECTS OF A STATIC PHASE ERROR ON TRANSMISSION OF A WHITE TO BLACK LUMINANCE TRANSITION.

In the previous section, a maximum amount of ISI as consequence of a static phase

error is derived. In order to have an idea of the relevance of this bound we will consider a more realistic situation in this section. This means that the whole transmission system of figure 9 is to be considered. Even a simple model of the BRE and the BRD will be introduced, to make a connection to a displayed luminance picture.

The HDTV camera (see figure 4) generates a picture existing of 1250 lines, 2:1 interlacing and 50 fields per second, often abbreviated as the 1250/50/2:1 scanning format. This signal is encoded in the BRE. For this purpose, the BRE distinguishes three different levels of motion: An area in the picture field with no motion is coded in the so called 80 ms branch, areas with slow motion are coded in the 40 ms branch and fast moving areas are coded in the 20 ms branch. This is done to reduce the information-stream in such a way that it can be transmitted over the 625/50/2:1 MAC-channel. The operations performed in the different branches are based on filtering and quincunx sub-sampling. It is beyond the scope of this report to explain the rather complex technique of HD-MAC encoding and decoding; a brief introduction can be found in [1] and [2]. We will focus our attention on the 80 ms branch, the *static mode.*

The 80 ms static mode is the most critical mode in the HD-MAC encoding process. It deals with non-moving areas in the picture, so noise and other disturbances are not 'hidden' in the action of a moving scene, but can be examined at ease on the display. Fortunately, the static mode is also the least complex mode: By means of diagonal filtering the data-rate is halved. The resulting effect of this technique for a non-moving scene is that two lines are transmitted as one by taking one sample from the first line, the next sample from the second line, the third from the first line again, and so on. The operation of the BRE in its most critical mode can thereby be described as what is called *line shuffling.* The inverse action of the BRD consists of

deshuffling the received lines.

The HDTV picture we are going to transmit consists of a white upper plane and a black lower plane: The first lines are all of high amplitude, the last lines all of low amplitude. The solitude white or black plane is not interesting: Even in the presence of a static phase error in the sampler it is likely that one level input signal comes out as one level output signal. What interests us is the transition from white to black. The BRE in the static mode makes of these two lines, of which one exists of a high amplitude and the other of a low amplitude, one line, consisting of a block shaped signal with alternating high and low amplitudes. This signal is the input of the E7 pre-emphasis filter. Under influence of the imperfect sampler it is to be expected that the sharp white-black transition will be 'flattened' into a somewhat more grey transition.

Suppose a white luminance level has the 8-bit number 240, corresponding with 0.5 V, and a black luminance level the 8-bit number 16 (=256-240), corresponding with -0.5 V (see chapter 2.1). Then, at the moment of the white-black transition, the signal at the input of the E7 pre-emphasis network is described as

$$x(kT) = \begin{cases} A = 0.5\ \text{V} & k \text{ is even} \\ -A = -0.5\ \text{V} & k \text{ is odd} \end{cases}$$

Using the software described in chapter 4, this signal is processed in the HD-MAC transmission chain, where the sampler is afflicted with a certain static phase error ΔT. The output of the E7 de-emphasis network will be an alternating signal with amplitude $\pm A^*$, where $A^* \leq A$. After the line-deshuffling operation performed in the BRD, the result is a "less white" line with amplitude A^*, and a "less black" line with amplitude $-A^*$. $A^* = 0$ corresponds to a neutral grey luminance level. Figure 16 shows the value A^* as a function of the static phase error ΔT. The amount of ISI follows directly from A^*.

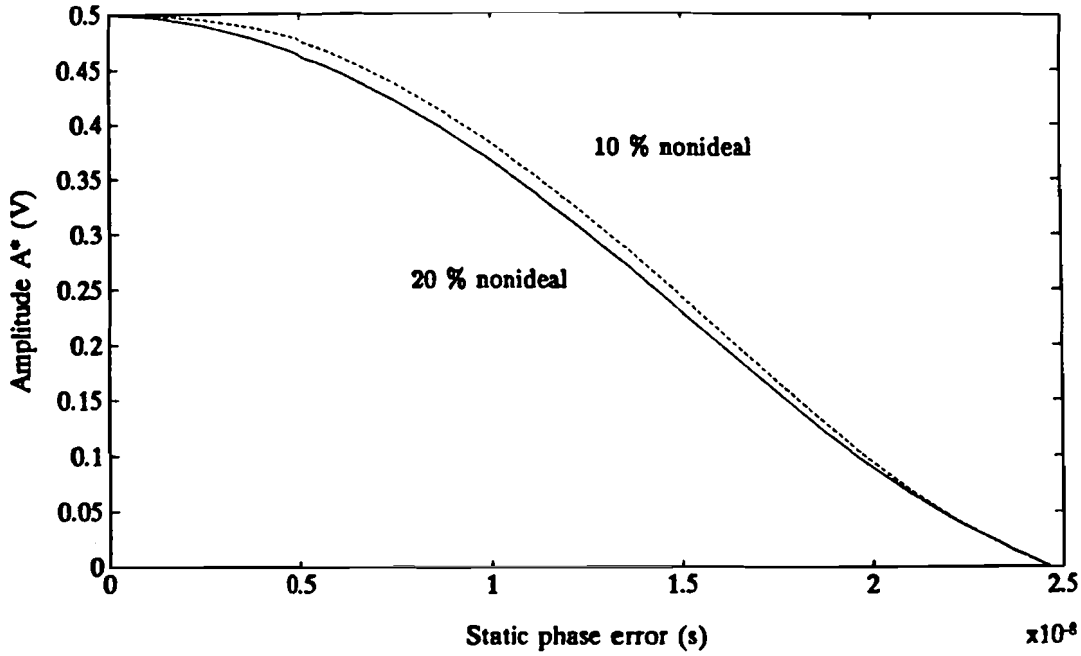

Figure 16: Absolute amplitude of the output of the E7 de–emphasis network in transmission of a white to black luminance transition.

As expected, at $\Delta T = T/2 \approx 25$ ns (in the middle of two sample instants) $A^* = 0$, corresponding to 100 % ISI. At $\Delta T = 10$ ns is $A^* \approx 0.38$ V for both Nyquist filters, corresponding to approximately 25 % ISI. Comparing this with the maximum amount of 90 %, respectively 120 % ISI for nonideal Nyquist filtering in figure 15, the result is quite gratifying. Note however, that for $\Delta T > 10$ ns the amount of ISI increases rapidly. An interesting aspect of figure 16 is that for the described input signal and the used system the 10 % nonideal Nyquist filter has a better performance than the 20 % nonideal Nyquist filter.

6.3. EFFECTS OF A STATIC PHASE ERROR ON THE AMPLITUDE SPECTRUM.

Thus far we have considered the influence of a random or static phase error in terms

of ISI. It means that the error caused by the sampler is expressed in percents of the original amplitude of the processed sample. Most of the analysis is done without respect to the characteristics of the signal; i.e. each sample was considered on its own and taken more or less independent of the rest of the samples. The other samples were only interesting in the way they might cause *maximum* ISI on our beloved sample, or, as in section 6.2, in the way they fitted in relation with a realistic HD-MAC picture.

A different approach to examine the effects of a phase error of the sampler on a signal is to investigate the changes in the frequency domain of this signal after missampling. It might be, for example, that high-frequency signals are more affected by an imperfect sampler than low-frequency signals. Since unequal gains in the amplitude spectrum can cause significant distortions on the display (note that the Nyquist filters also have been designed using a *equiripple* algorithm, see appendix A), it is worth investigating it. Section 6.3.1 shows the motive for further investigation: A simple example is used to reveal the effect of static phase error on a signal of frequency 0 or $\frac{1}{2T}$. Section 6.3.2 provides a more fundamental analysis of the problem.

Both section 6.3.1 and section 6.3.2 consider a cascade of a DAC and an ADC (sampler). The DAC is assumed to be ideal, while the sampler is afflicted with a certain static phase error. Let $x(kT)$ be the time-discrete signal at the input of the DAC and $x(t)$ the time-continuous output of the DAC. $x(t)$ is sampled at timing instants $t_s = kT + \Delta T$, where ΔT denotes the static phase error of the sampler. The output of the sampler is denoted as $z(kT)$, so

$$z(kT) = x(kT + \Delta T)$$

6.3.1. Effects for a signal of frequency 0 or $\frac{1}{2T}$

Let the input of the DAC be the time–discrete signal

$$x(kT) = C.\cos(2\pi f_b kT) \tag{6.3}$$

where C is any real constant. Suppose the sampler has a static phase error ΔT, then the output of the sampler is

$$z(kT) = C.\cos(2\pi f_b kT + 2\pi f_b \Delta T) \tag{6.4}$$

Now, consider the input signal $x(kT)$ with $f_b = 0$. We then have

$$x(kT) = z(kT) = C$$

$x(kT)$ is a signal that only has a zero–frequency component in its amplitude spectrum. We see that, no matter what ΔT can be, the signal is correctly restored and nothing in the amplitude spectrum is changed.

At first sight, the example given for $f_b = 0$ is rather trivial. But now look at the other extreme: Suppose $f_b = \frac{1}{2T}$, so

$$\begin{aligned} x(kT) &= C.\cos(k\pi) \\ &= \begin{cases} C & k \text{ is even} \\ -C & k \text{ is odd} \end{cases} \end{aligned}$$

For $z(kT)$ it then follows:

$$\begin{aligned} z(kT) &= C.\cos(k\pi + \pi\tfrac{\Delta T}{T}) \\ &= C.\cos(k\pi)\cos(\pi\tfrac{\Delta T}{T}) - C.\sin(k\pi)\sin(\pi\tfrac{\Delta T}{T}) \\ &= \cos(\pi\tfrac{\Delta T}{T}) \,.\, x(kT) \end{aligned} \tag{6.4}$$

If $|X(f)|$ denotes the amplitude spectrum of $x(kT)$ ($X(f)$ obtained by the FFT) and $|Z(f)|$ the amplitude spectrum of $z(kT)$, equation (6.4) implies the following relationship:

$$|Z(f)| = \cos(\pi\tfrac{\Delta T}{T}).|X(f)| \tag{6.5}$$

if $|\Delta T| \leq \frac{T}{2}$ (which is always possible).

Equation (6.5) shows that for the particular case $f_b = \frac{1}{2T}$ the influence of the static

phase error cannot be ignored. The question arises how the behaviour of the amplitude spectrum will be for other frequencies than $f_b = 0$ or $f_b = \frac{1}{2T}$ as function of the static phase error ΔT.

6.3.2. Theoretical analysis.

Taking for $x(kT)$ the time–discrete signal of equation (6.3) has the advantage that the amplitude spectrum $X(f)$ for $0 \leq f \leq \frac{1}{2T}$ (using the FFT) consists of approximately one frequency component at $f = f_b$. Other frequencies in the spectrum have a very low amplitude in comparison to this frequency $f = f_b$. How small these other amplitudes are depends on the way the FFT is taken. The FFT of $x(kT)$ is defined as [10]:

$$X(n) = \sum_{k=0}^{N-1} x(kT)\ W_N^{-kn} \qquad n = 0,1,..,N-1 \qquad (6.6)$$

where $W_N = \exp(j\frac{2\pi}{N})$

Equation (6.6) is called a N–point FFT. The greater N, the more the amplitude spectrum will approach the shape of a delta–function with its peak at $f = f_b$. $X(n)$ can be expressed in terms of f by substituting $n = f.N.T$.

Following the previous argumentation the signal $x(kT)$ given by equation (6.3) is the right signal to investigate the influence of the erroneous sampler at one frequency. More specific: If the sampler is afflicted with a static phase error ΔT and $|X(f)|$ is the amplitude spectrum of $x(kT)$, the amplitude of $|Z(f)|$ at $f = f_b$ shows us any possible unwanted gains.

Computing $G(n) = \left|\frac{Z(n)}{X(n)}\right|$ using equation (6.6) is a hard job for large N. However, for the particular case $x(kT) = \cos(2\pi f_b kT + 2\pi f_b \Delta T)$ the expression for $|Z(n)|$ can be written in a simple form. The neat derivation of this expression is found with help of J. van Wamel and shown in appendix E. The result is:

$$|Z(n)|^2 = \frac{1}{4}\left\{ A^2 + B^2 + 2AB.\cos\left[(\alpha+\beta)\frac{N-1}{2} + 4\pi f_b \Delta T\right] \right\} \qquad (6.7)$$

where $n = 0,1,..,N-1$

$$\alpha = 2\pi(f_b T - \tfrac{n}{N})$$

$$\beta = 2\pi(f_b T + \tfrac{n}{N})$$

$$A(n) = \frac{\sin(\frac{\alpha N}{2})}{\sin(\frac{\alpha}{2})}$$

$$B(n) = \frac{\sin(\frac{\beta N}{2})}{\sin(\frac{\beta}{2})}$$

$|X(n)|^2$ is obtained by substituting $\Delta T = 0$ in equation (6.7). Equation (6.7) enables us to derive an expression for $G(n)$. But, what we are really interested in is the behaviour of $G(n)$ at $f = f_b$. Substituting $n = f_b.N.T$ in equation (6.7) the expression reduces to

$$|Z(n)|^2_{n=f_b.N.T} = \frac{1}{4}\left\{ A^2 + B^2 + 2AB.\cos\left[(\alpha+\beta)\frac{N-1}{2} + 4\pi f_b \Delta T\right]\right\} \qquad (6.8)$$

where $n = 0,1,..,N-1$

$$\alpha = 0$$

$$\beta = 4\pi f_b T$$

$$A = N$$

$$B = \frac{\sin(2\pi f_b NT)}{\sin(2\pi f_b T)}$$

$G(f_b) = G(n)\big|_{n=f_b NT}$ follows directly from equation (6.8). Note that for a certain frequency f_b the result $G(f_b)$ is not only dependent on ΔT, but also on N, the length of the FFT. For $\Delta T = 2$ ns and $N = 2048$ or $N = 8192$ $G(f_b)$ is shown in figure 17.

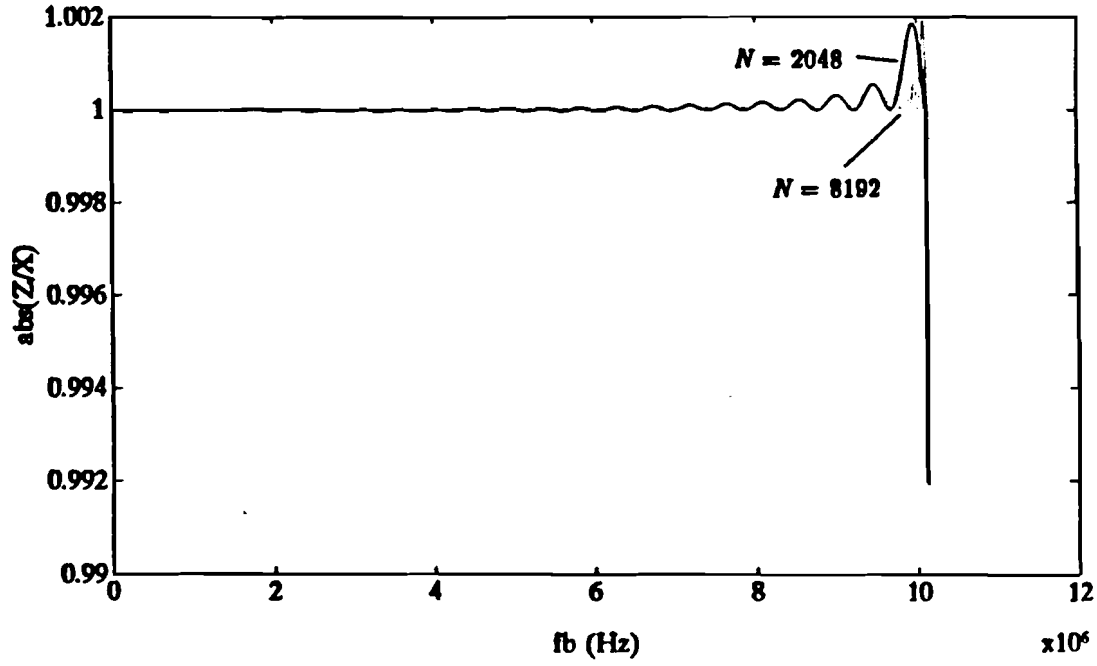

Figure 17: $G(f_b)$ for $\Delta T = 2$ ns and $N = 2048$ or $N = 8192$.

In figure 17 it can be seen that $f_b = \frac{1}{2T}$ is a limit case: The amplitude spectrum remains (approximately) flat up to $f_b = \frac{1}{2T}$. At $f_b = \frac{1}{2T}$ the amplitude spectrum behaves as given in equation (6.5). Note that $G(f_b)$ is a continuous function, so the discontinuity in figure 17 near $f_b = \frac{1}{2T}$ is fully due to the accuracy of the computation of the plot. An interesting aspect of figure 17 is that the curve for $N = 8192$ shows less distortion for $G(f_b)$ than the curve for $N = 2048$. For larger N, the effect is even more surprising. Figure 18 shows $G(f_b)$ for $N = 65536$. In this figure, $G(f_b)$ is approximately 1.

The results can be interpreted as follows. N is the length of the FFT: i.e. the greater N, the more samples of $x(kT)$ and $x(kT)$ are involved in the process, the less the distortion caused by the erroneous sampler is. What does this imply for ordinary signals in a transmission situation?

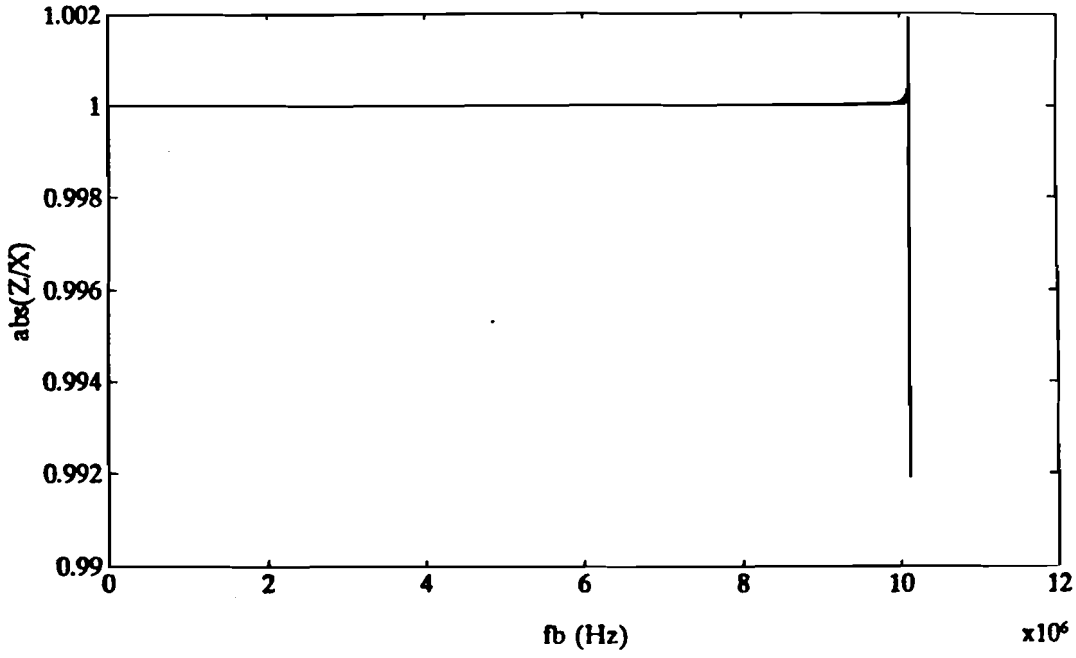

Figure 18: $G(f_b)$ for $\Delta T = 2$ ns and $N = 65536$.

Since variable N is only a mathematical help in deriving the FFT, no relationship to the real world seems to exist. However, considering N to be the duration of a frequency component in a signal, figures 17 and 18 have a meaning. When a high–frequency component is present for a long period of time in a signal (N is high) the effect of the erroneous sampler is less than when the high–frequency component is present for a short period of time (N is low). The actual duration is calculated by multiplying N with T, the sampling time.

6.4. CONCLUSIONS.

The analysis for the maximum amount of ISI when the sampler has a static phase error shows that the sampling instant must be accurate within nanoseconds. A static phase error of a few nanoseconds can cause tens of percents of ISI. Further, for static phase errors less than 2 ns there is no difference between the 20 % and the 10 % nonideal Nyquist filtering in the maximum amount of ISI. The difference between

ideal- and nonideal Nyquist filtering is constant and amounts to approximately 15 % ISI, depending on the roll-off factor of the Nyquist filter.

The case study shows that the amount of ISI to be expected in a realistic situation is much lower than the bound given for the maximum amount of ISI. For a static phase error $\Delta T = 10$ ns we have an amount of ISI of 25 %, for both the roll-off factor of 20 % and the roll-off factor of 10 %.

The distortions in the amplitude spectrum due to a static phase error are seen to be minimal. Only for frequencies of $\frac{1}{2T}$ (a limit case) the distortion is significant. High-frequency components are more sensitive to a static phase error than low-frequency components, but the longer a frequency component is present in a signal, the less the average distortion will be.

7. Noise power spectral density analysis.

In the previous two chapters the effects of an imperfect sampler have been investigated. In this chapter we will examine the effects of channel noise. We will focus on the elements of the receiver as given in the receiver-side of figure 9, i.e. the FM-demodulator, E1 de-emphasis, ADC (sampler), 1/2 Nyquist, subsampler and the E7 de-emphasis network. Also the alternative E8 de-emphasis, as described in chapter 2.1, will be used. The Nyquist filtering is done with a roll-off factor of 10 %. Furthermore, the noise at the input of the receiver is assumed to be additive, Gaussian and white, the noise power is denoted as N (see also chapter 3.4). In the following the output noise psd is examined for different ways of de-emphasis filtering. By definition, the noise power spectral density (psd) shows the power of the noise at various frequencies. For the input white noise, the noise power is equal for each frequency, but the receiving process will make certain frequencies more sensitive to noise than others, as we have already seen in chapter 3.

In the first section a theoretical analysis for the system without E7 or E8 de-emphasis will be given. The second and the third part of this chapter present the simulations and the results for the noise psd for the complete system. In section four these results will be interpreted and discussed.
The human eye is more sensitive at low frequencies than at high frequencies: In the fifth section a connection to the visibility of the noise on the HDTV display will be made.

7.1. THEORETICAL ANALYSIS.

The aim of this section is to derive an expression for the noise psd at the output of the receiver. For this we need the transfer function of each device of the receiver-side

in figure 9. Unfortunately, the E7 or E8 de-emphasis filter is a nonlinear network, so it is not possible to describe it in terms of frequency transfer functions. The de-emphasis filtering will therefore be omitted from consideration here; its effect on the noise psd is examined in the simulations part of this chapter. Also the function of the samplers is left out in the derivations (for the time being). The resulting system is given in the block diagram of figure 19.

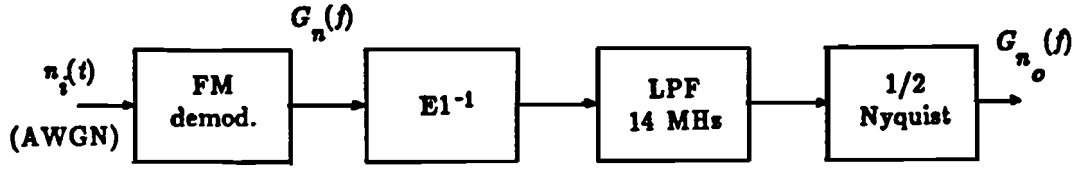

Figure 19: Parts of the receiver as used for the derivation of the output noise psd.

In accordance with chapter 3, the output noise psd of the FM-modulator with additive Gaussian white input noise is given by

$$G_n(f) = kf^2$$

with k as in equation (3.16). The transfer function of the E1^{-1} linear de-emphasis filter is (see equation (2.4)):

$$H_{\text{E1}^{-1}}(f) = \frac{1}{A} \cdot \frac{1+j\,(f/f_2)}{1+j\,(f/f_1)}$$

The low pass filter with cut-off frequency at 14 MHz is assumed to be ideal.

7.1.1. Ideal Nyquist filtering.

For ideal Nyquist filtering, the output noise psd is given by the equation

$$G_{n_o}(f) = G_n(f).|H_{\text{E1}^{-1}}(f)|^2.|H_{\text{LPF}}(f)|^2.|P_{\frac{1}{2}}(f)|^2$$

where $P_{\frac{1}{2}}(f)$ is the transfer function of the ideal half Nyquist filter. $|P_{\frac{1}{2}}(f)|^2$ is

expressed as (Note that, as in equation (3.2), the factor T is omitted)

$$|P_{\frac{1}{2}}(f)|^2 = \begin{cases} 1 & |f| < \frac{1}{2T}(1-\alpha) \\ \cos^2\left\{ \frac{\pi T}{2\alpha}(|f| - \frac{1}{2T}(1-\alpha)) \right\} & \frac{1}{2T}(1-\alpha) \le |f| \le \frac{1}{2T}(1+\alpha) \\ 0 & |f| > \frac{1}{2T}(1+\alpha) \end{cases}$$

The low-pass filter has a flat frequency response up to $\frac{1+\alpha}{2T}$, so

$$G_{n_o}(f) = \begin{cases} k(f/A)^2 \frac{1+(f/f_2)^2}{1+(f/f_1)^2} & |f| < \frac{1}{2T}(1-\alpha) \\ k(f/A)^2 \frac{1+(f/f_2)^2}{1+(f/f_1)^2} \cos^2\left\{ \frac{\pi T}{2\alpha}(|f| - \frac{1-\alpha}{2T}) \right\} & \text{elsewhere} \\ 0 & |f| > \frac{1}{2T}(1+\alpha) \end{cases} \qquad (7.1)$$

Figure 20 shows $G_{n_o}(f)$ for $\alpha = 0.1$ and an input signal-to-noise ratio $C/N = 18$ dB.

7.1.2. Nonideal Nyquist filtering.

The nonideal Nyquist filter as specified in appendix A is a digital filter. To compare $G_{n_o}(f)$ for the nonideal filter with equation (7.1), we rewrite equation (2.3) or (A-2) in the continuous function equivalent

$$p'_{\frac{1}{2}}(t) = \sum_{k=0}^{N-1} c_k \, \delta(t - k\tfrac{T}{2}) \qquad (7.2)$$

where $p'_{\frac{1}{2}}(t)$ denotes the impulse response of the nonideal half Nyquist filter, N the number of taps of the filter and c_k $(0 \le k \le N-1)$ the gains of the taps. $\delta(t)$ is the ordinary delta (Dirac) function given by

$$\delta(t) = \begin{cases} 1 & t = 0 \\ 0 & t \neq 0 \end{cases}$$

$p'_{\frac{1}{2}}(t)$ can be Fourier transformed into

$$P'_{\frac{1}{2}}(f) = \sum_{k=0}^{N-1} c_k \, e^{-j\pi f k T} \qquad (7.3)$$

since $\delta(t-t_0) \longleftrightarrow e^{-j2\pi f t_0}$. Again, $G_{n_o}(f)$ follows from

$$G_{n_o}(f) = G_n(f) \cdot |H_{E1\text{-}1}(f)|^2 \cdot |H_{LPF}(f)|^2 \cdot |P_{\frac{1}{T}}^{!}(f)|^2$$

The amplitude spectrum $|P_{\frac{1}{T}}^{!}(f)|^2$ is derived as follows:

$$\begin{aligned}
|P_{\frac{1}{T}}^{!}(f)|^2 &= \left| \sum_{k=0}^{N-1} c_k \, e^{-j\pi fkT} \right|^2 \\
&= \left| \sum_{k=0}^{N-1} c_k \left(\cos(\pi fkT) - j \sin(\pi fkT) \right) \right|^2 \\
&= \left(\sum_{k=0}^{N-1} c_k \cos(\pi fkT) \right)^2 + \left(\sum_{k=0}^{N-1} c_k \sin(\pi fkT) \right)^2 \\
&= \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} c_k c_i \cos(\pi fkT) \cos(\pi fiT) + \\
&\qquad \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} c_k c_i \sin(\pi fkT) \sin(\pi fiT) \\
&= \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} c_k c_i \cos\left(\pi f(k-i)T \right) \qquad (7.4)
\end{aligned}$$

The output noise psd is then expressed as:

$$G_{n_o}(f) = \begin{cases} k(f/A)^2 \, \frac{1+(f/f_2)^2}{1+(f/f_1)^2} \sum_{k=0}^{N-1} \sum_{i=0}^{N-1} c_k c_i \cos\left(\pi f(k-i)T \right) & |f| \leq 14 \text{ MHz} \\ 0 & |f| > 14 \text{ MHz} \end{cases} \qquad (7.5)$$

For the 10 % digital Nyquist filter with 31 taps and $C/N = 18$ dB equation (7.5) is shown in figure 20.

It can be seen in figure 20 that the noise psd for the system with nonideal Nyquist filtering oscillates around the noise psd for the system with ideal Nyquist filtering. Despite the E1 de-emphasis filtering the noise psd still has a quadratic shape, due to the quadratic function of the noise psd at the output of the FM-demodulator. This indicates that higher frequencies are more sensitive to noise than lower frequencies at the output of the receiver. Figure 21 shows the curves of figure 20 in dB-scale. In this figure the difference between the two curves is very small.

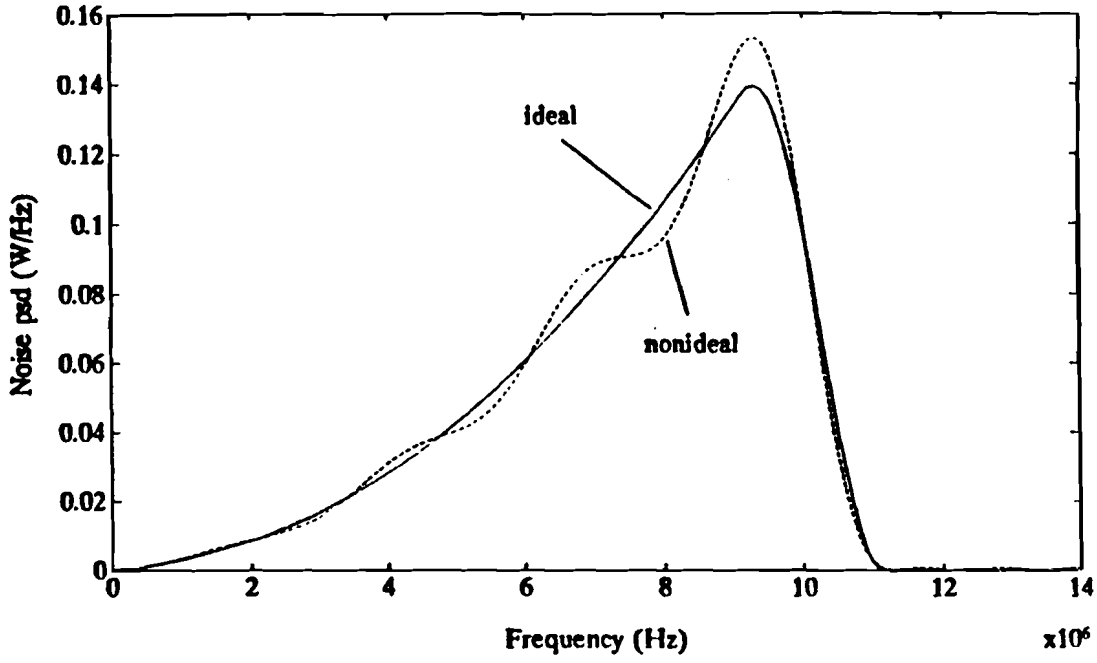

Figure 20: Noise psd's for ideal– and nonideal Nyquist filtering with $\alpha = 0.1$ and $C/N = 18$ dB

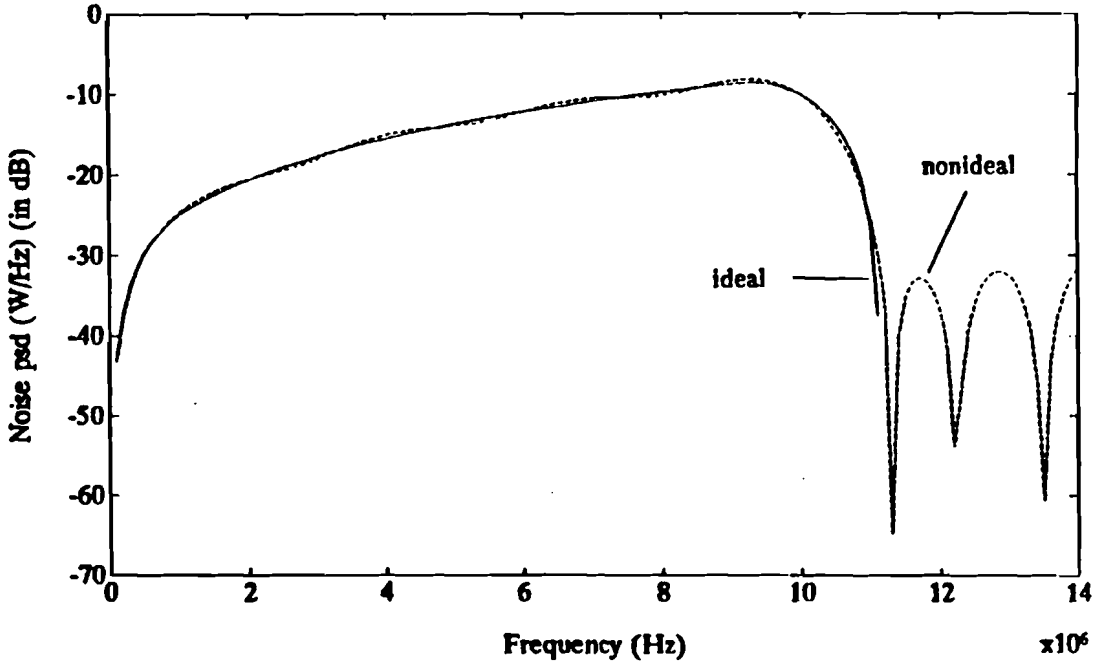

Figure 21: As figure 20, in dB–scale

7.2. DESCRIPTION OF THE SIMULATION METHOD.

Since it is not easy to derive an analytical expression for the noise psd at the output of the receiver, simulations have been carried out. The software described in chapter 4 is used. Consider for the simulations the receiver–side of figure 9: The noise at the input of the PLL is additive, Gaussian and white (depending on C/N); this noise is processed through the receiver up to and including the E7 (or E8) de–emphasis network. From the noise at the output of the de–emphasis network the psd can be computed.

To be sure that the results of such a simulation make sense, many noise samples must be generated, so that the statistical properties of the input noise are indeed Gaussian. However, since the software processing of the PLL, the E1 de–emphasis and the LPF takes a lot of time (they are the *non*–digital devices, see appendix B) this simulation might exceed one's patience. To overcome the problems of weeks of simulations we make use of a *noise filter*. This noise filter transforms noise with a Gaussian distribution into noise with the distribution at the output of the sampler, thus avoiding the need for simulating long analog signals. The receiver side of figure 9 can (with use of the noise filter) for the simulations be simplified as shown in figure 22.

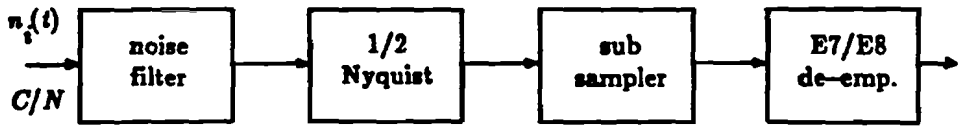

Figure 22: Block diagram of the receiver side as used in the noise psd simulations.

The noise filter is a 21 taps digital filter with a sample frequency of 40.5 MHz. The gain coefficients of the taps are specified in appendix F. The filter expects samples at a sample frequency of 40.5 MHz from a random process with a Gaussian distribution and returns a close approximation of the noise samples as if they were processed in the PLL, the E1 de-emphasis, LPF and the ADC.

With the software describing the elements in the block diagram of figure 22 the following procedure is carried out:

1. Generate a noise-vector from a normal distribution $N(\mu=0,\sigma)$, where σ is depending on C/N. This vector contains about 1000 samples of the noise process.
2. Process the noise-vector through the system given in figure 22.
3. Take the Fast Fourier Transform (FFT) of the final response and determine the noise psd sample by multiplying it with its conjugate.
4. Repeat step 1-3 several (2000-6000) times to approximate the normal distribution as closely as possible and take the average of the results.

The total number of processed noise samples is about 2-6 million, enough to get a thorough impression of the noise psd. A listing of the program for the described simulation can be found in appendix D.

7.3. SIMULATION RESULTS.

The prescribed procedure has been carried out for the system of figure 22 without de-emphasis filtering, with E7 and with E8 de-emphasis filtering for $C/N = 14$, 18 and 22 dB. In each simulation about 6 million noise samples were generated. Figure 23 shows the noise psd (in dB) for the system of figure 22 with E7 de-emphasis filtering. The three curves are for $C/N = 14$, 18 and 22 dB. It can be seen that an increase of 4 dB in C/N results in a decrease of the noise psd of 4 dB.

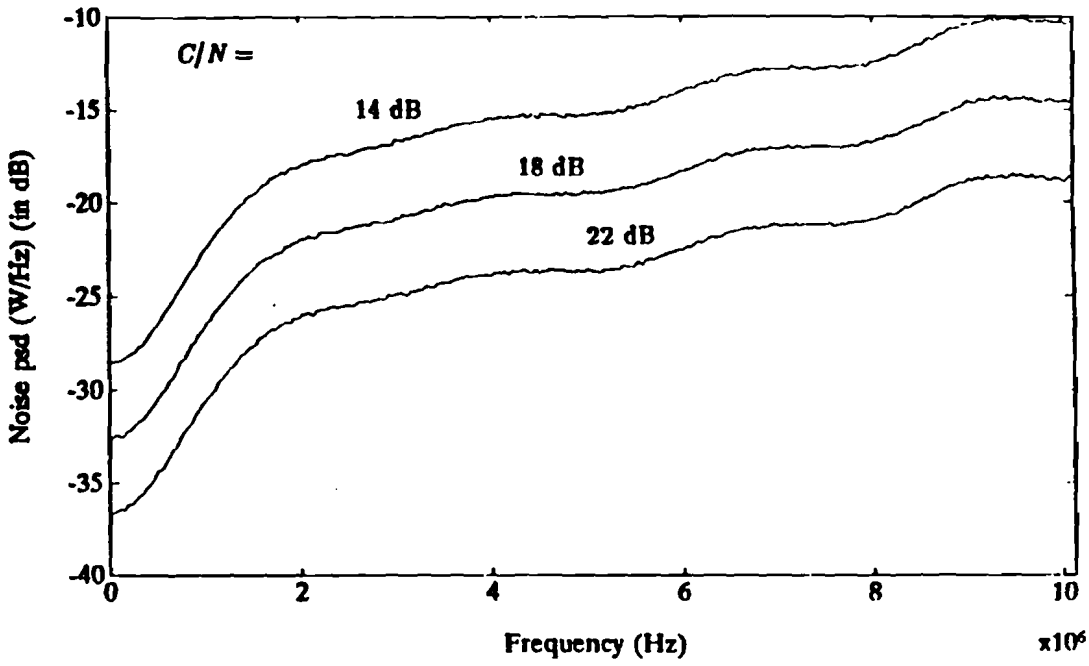

Figure 23: Noise psd's for the receiver with E7 de–emphasis.
C/N = 14,18 and 22 dB.

Figures 24 and 25 show the noise psd for the three different receiver configurations: Without de–emphasis, with E7 and with E8 de–emphasis. Figure 25 is the same as figure 24, only represented in dB's. In these figures the input signal–to–noise ratio C/N is always 18 dB.

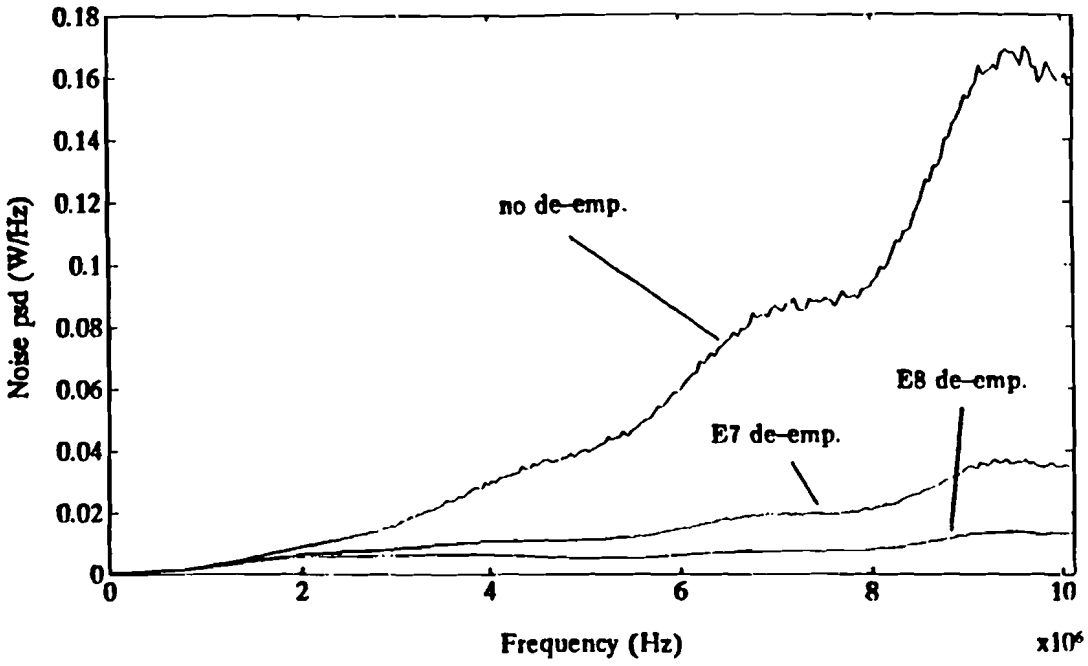

Figure 24: Noise psd for the receiver without de–emphasis, with E7 de–emphasis and with E8 de–emphasis. $C/N = 18$ dB.

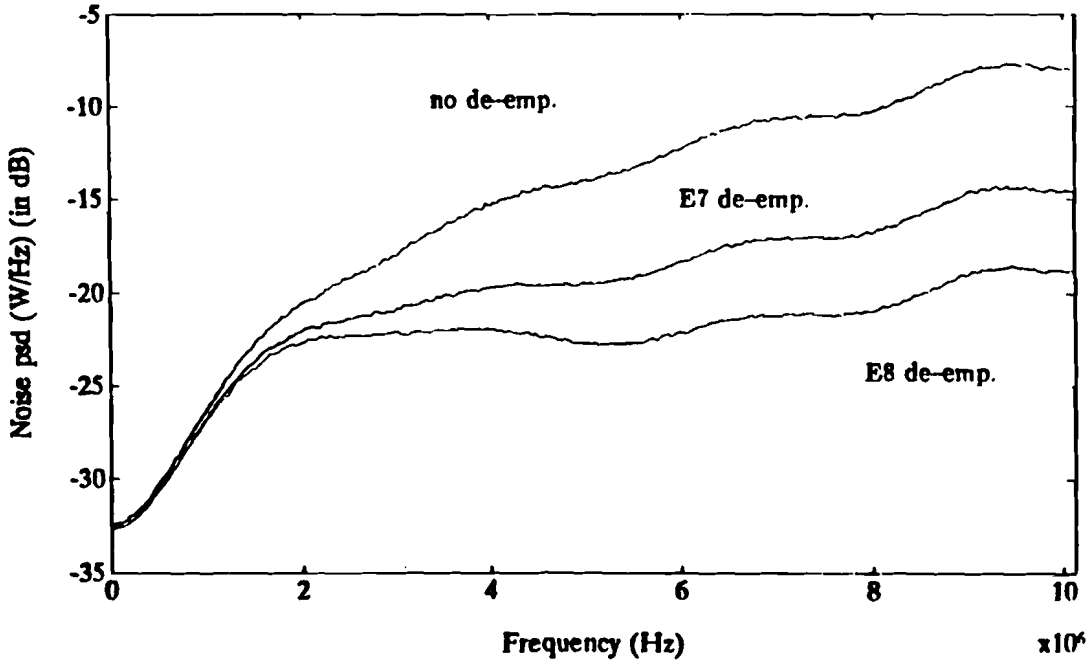

Figure 25: As figure 24, the noise psd's in dB–scale.

The curve in figure 24 for the receiver without de–emphasis filtering matches with our theoretical analysis for nonideal Nyquist filtering in chapter 7.1, figure 20. The operation of the de–emphasis filters can be clearly seen in figure 25: Low frequency signals are not affected at all, but at higher frequencies the filters are able to decrease the noise power significantly. For the E7 filter, the difference with no de–emphasis filtering is at most 6 dB. For the E8 de–emphasis filter the difference with no de–emphasis filtering is 11 dB.

7.4. DISCUSSION OF THE RESULTS.

The great differences in the noise psd for the various receivers seem quite shocking. Consider, for example, transmission of a signal of frequency 9 MHz. This signal enters the transmitter as a signal of frequency 9 MHz and leaves the receiver as a signal of frequency 9 MHz, independent of the sort of pre– and de–emphasis that is used. This is what designing is all about: Reliable transmission of signals. But, inevitably the signal will be disturbed by noise. For HDTV (and most other systems), the quantity of the disturbance is often expressed in the input signal–to–noise ratio C/N. However, figure 25 shows that the influence of noise at a certain C/N is dependent of the way the receiver is designed. Not just a little, but up to 11 dB. Since the signal remains unchanged, the output signal–to–noise ratio S_o/N_o may differ 11 dB according to the way the receiver is designed.

This seems quite shocking, but only if we forget the fact that the de–emphasis filter is a *nonlinear* device. One of the consequences is that additive noise cannot be considered additive anymore. In other words: The processing of the noise in the de–emphasis filter is dependent of the signal that is carrying the noise. As we have seen in chapter 2.1, the de–emphasis filter has its maximum effect on the low amplitude components of high frequency signals. In our simulation, the signal was

absent and system noise has generally a low amplitude. The gain of 6 or 11 dB for E7 or E8 de-emphasis filtering at high frequencies is therefore the absolute maximum gain that can be obtained. Transmission of a real signal, disturbed by noise, will diminish the effect of nonlinear de-emphasis filtering.

To check the operation of the de-emphasis filter in transmission of a signal disturbed by noise, we adapt the prescribed simulation procedure. A signal is transmitted through the system up to the de-emphasis filter. Since the rest of the receiver is linear, this can also be done for the noise, independent of the characteristics of the signal. Before processing in the de-emphasis network, the signal and the noise are added. The result of this processing is the receiver output signal, disturbed by noise. The desired signal, i.e. the noiseless output signal, is subtracted from this result,so that the noise at the output of the receiver is obtained. From this noise, the psd can be determined as described before. The new simulation is executed for two input signals: A sinusoid of frequency 5 MHz with a top-amplitude of 0.3 V (high amplitude) and a sinusoid of frequency 5 MHz with a top-amplitude of 0.05 V (low amplitude). The result of the simulation for $C/N = 18$ dB and E7 de-emphasis filtering are shown in figure 26. In both simulations about 2 million noise samples were generated.

Figure 26 shows us that the noise psd in the absence of a signal is indeed a lower bound. With transmission of a signal the noise psd lies above this lower bound and even exceeds the noise psd of the receiver where no de-emphasis is used. For certain signals, the de-emphasis filtering may thus diminish the output signal-to-noise ratio S_o/N_o. In most practical situations the noise psd curve will lie roughly between the lower bound curve and the curve for the noise psd of the receiver without de-emphasis, thus accomplishing a gain of about 3 dB for high frequencies. For the E8 de-emphasis filter the gain would be 6 dB by the same reasoning.

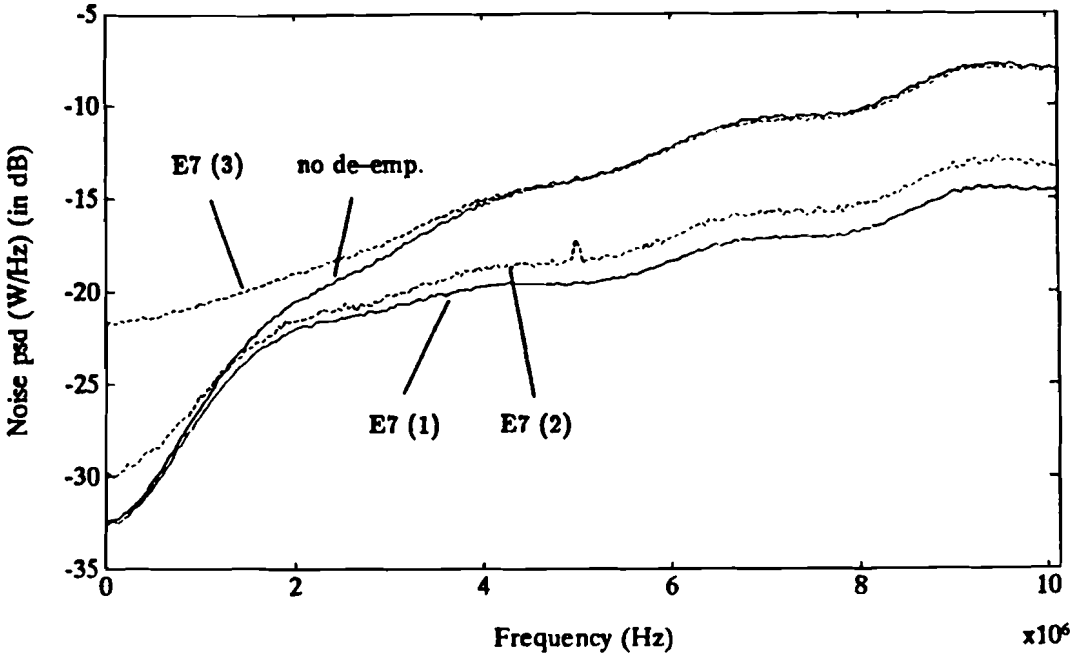

Figure 26: Noise psd for the receiver without de–emphasis and for the receiver with E7 de–emphasis with three input signals: No input signal (1), low amplitude (2), high amplitude (3).

7.5. APPROXIMATION TO THE EYE–SENSITIVITY.

In the previous sections we have examined the influence of white noise at the receiver input for various frequencies at the receiver output. The result has always been shown in the noise psd; for various receiver designs, input signal–to–noise ratio's and input signals. However, the only question that really interests us is: What is visible on the the HDTV display? In this section we neglect the influence of the BRD and assume that the noise psd's stated in the previous sections are the noise psd's appearing on the display. This does not settle the answer, for the human eye notices low–frequency noise immediately, while high–frequency noise can scarcely be seen. A lot of research has been carried out in the field of the human visual system, resulting in various models and noise weighting functions. Due to the complex processing of the video

signal in the BRE and BRD, which affects both space- and time dimensions, it is not easy to derive a simple noise weighting model for the HD-MAC system, see for example [11] or [12], as it is for conventional TV. It is however far beyond the scope of this report to consider all difficulties arising in deriving such a model.

To obtain an impression of the sensitivity of the human visual system to the noise psd's of the previous sections a simple 1-dimensional model is used. This model is given as a digital video weighting filter; a 41 taps transversal filter operating at a sample frequency of 20.25 MHz, representing the sensitivity of the human visual system to various frequencies. The specifications of this filter are given in appendix G.

Regarding the video weighting filter as a supplementary device after the de-emphasis filter, the psd of the output of the system gives us the relative sensitivity of the human eye to the input noise. Let $VW(f)$ be the transfer function of the video weighting filter and $G_{n_o}(f)$ the noise psd at the output of the receiver. Then the new psd $G_{eye}(f)$ for the human eye is expressed as

$$G_{eye}(f) = G_{n_o}(f).|\,VW(f)|^2 \qquad (7.6)$$

The dimension of equation (7.6) is of course W/Hz, though this is rather meaningless: What is the definition of 1 Watt in relationship to the human eye and to the processing in the brains of what is seen? What interests us is the shape of the new obtained psd. We could therefore label the new psd as 'the relative disturbance people have while looking at the HDTV display'.

Applying equation (7.6) to the results derived in chapter 7.3, we obtain figures 27 and 28. These figures show the noise psd's of figures 23 and 25, modified by equation (7.6).

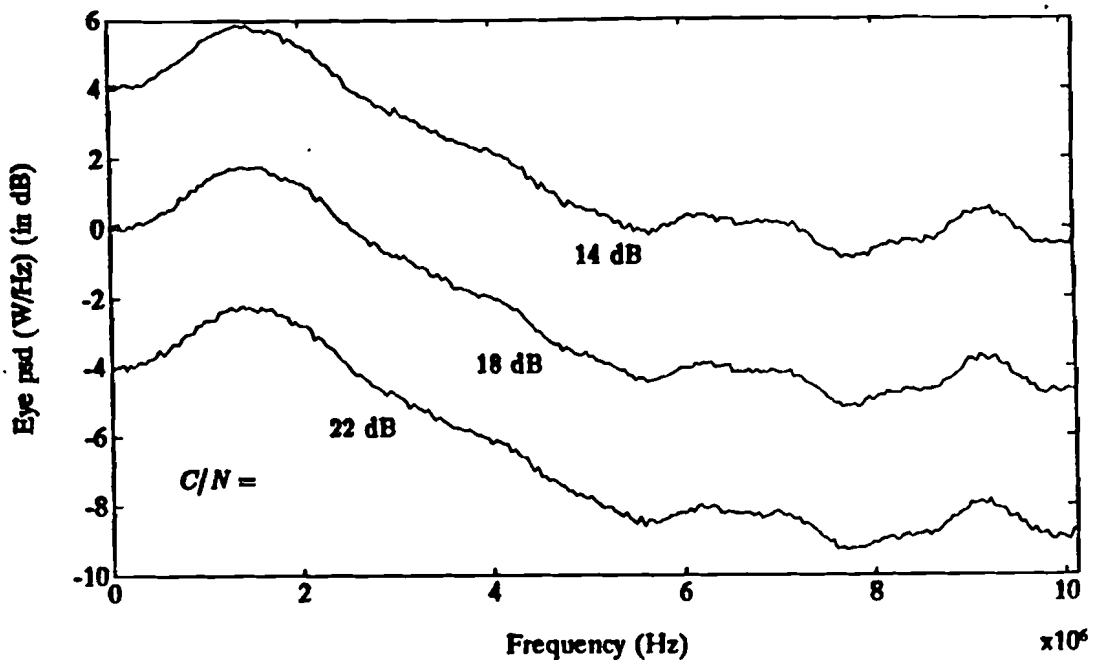

Figure 27: Noise psd's of figure 23 after processing in the video weighting filter (normalised).

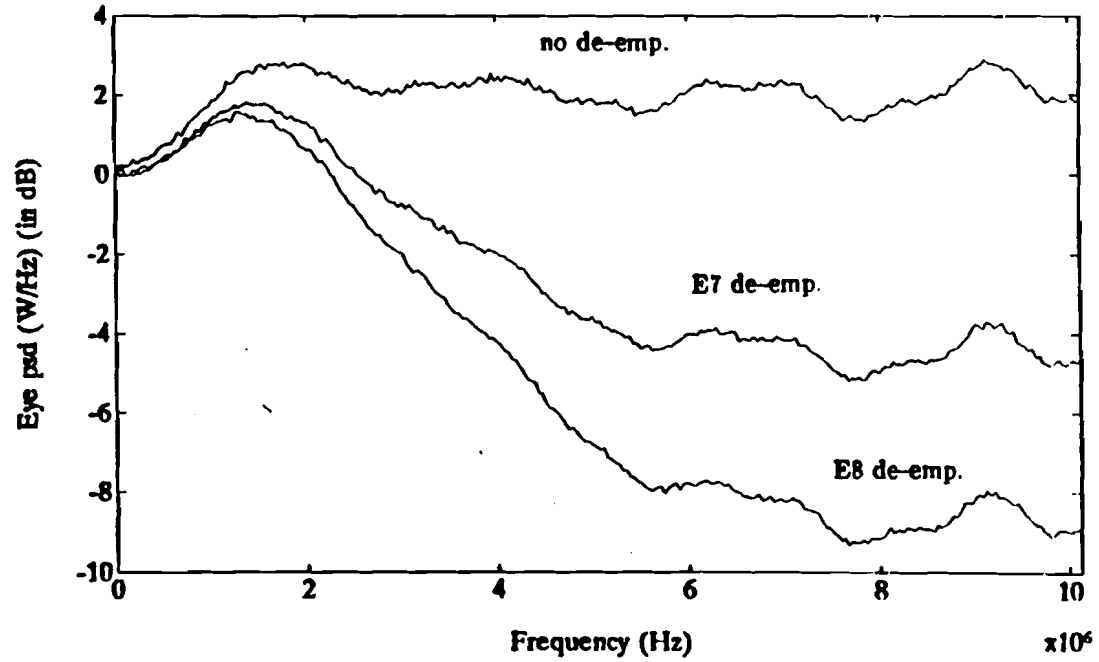

Figure 28: Noise psd's of figure 25 after processing in the video weighting filter (normalised).

Figure 27 shows that the difference between sensitivity for low frequencies and high frequencies is compensated. In figure 23, the noise psd covers a range of 18 dB; in figure 27 this is reduced to approximately 7 dB. The severe noise power for high frequencies in figure 23 causes only few disturbance for the eye; the critical frequency for which noise is to be seen is about 1.5 MHz. In figure 28 the operation of the de-emphasis filtering comes out clear. When using a receiver without de-emphasis filtering the noise is about equally visible at all frequencies. The E7- and E8-network come into operation at frequencies above approximately 2 MHz, where they succeed in reducing the noise power sensitive to the eye. Again the covering range of the psd is diminished; from 18 dB (E7) respectively 14 dB (E8) in figure 25, to 7 dB (E7) respectively 11 dB (E8) in figure 28. Note that figures 27 and 28 show the eye psd for noise in absence of a signal, so the remarks made in chapter 7.4 are still valid.

7.6. CONCLUSIONS.

The noise psd's at the output of the three receivers (no de-emphasis, E7 or E8 de-emphasis) all show a higher noise power for high frequencies than for low frequencies. Using E7 de-emphasis filtering in the receiver gives a maximum gain of 6 dB in S_o/N_o for high frequency signals, using the E8 de-emphasis filter the gain is at most 11 dB for high frequency signals. The actual gain that can be practically obtained is very much dependent of the characteristics of the input signal at the receiver.

Since the human visual system is more sensitive to low frequency signals than to high frequency signals, noise at low frequencies is seen to be most disturbing. The critical frequency for all receiver designs is about 1.5 MHz. The gain when using de-emphasis filtering is obtained for frequencies above 2 MHz; the disturbance that can be seen by a human observer is significantly reduced for these frequencies, up to 7 dB for E7

de-emphasis filtering or 11 dB for E8 de-emphasis filtering in comparison with no de-emphasis filtering.

8. Conclusions.

Nyquist filtering equally shared between transmitter and receiver is shown to be the best configuration for a system using frequency (de)modulation. A very close suboptimum is obtained when full Nyquist filtering is carried out in the receiver, but no hardware can be saved by doing so. A significant degradation in performance is the consequence of a full Nyquist filtering in the transmitter. These results are independent of the roll-off factor of the Nyquist filter.

The analysis of the HD-MAC system with an erroneous sampler (ADC) has been carried out in two parts: First, effects of a random phase error in the sampling instant; second, effects of a static phase error in the sampling instant. Both analyses for a maximum amount of intersymbol interference (ISI) show the same results:

- The sampling instant is very critical: small deviations in the sampling instant give rise to a maximum ISI distortion of tens of percents. Therefore, the sampling instant must be accurate to within nanoseconds.
- Nyquist filtering with a roll-off factor of 10 % is more critical than Nyquist filtering with a roll-off factor of 20 %, where the difference is somewhat stronger in case of a static phase error.
- The difference between ideal- and nonideal Nyquist filtering is constant in the maximum amount of ISI. Nonideal Nyquist filtering produces an approximately 20 % higher maximum amount of ISI than ideal Nyquist filtering.
- Nonideal Nyquist filtering can even in correct operation of the sampler produce a maximum amount of ISI of 20 %.
- For a random or a static phase error with a deviation smaller than 2 ns there is almost no difference between nonideal Nyquist filtering with a roll-off factor of 10 % or 20 % in the maximum amount of ISI.

Furthermore,

- A sampler afflicted with a random phase error can be decomposed in terms of a correct sampler and an addition of Gaussian white noise.
- A more realistic example for a system where the sampler has a static phase error shows that the difference in performance between nonideal Nyquist filtering with a roll-off factor of 20 % or 10 % is very small. The amount of ISI in this example is approximately $\frac{1}{4}$ of the maximum amount of ISI.
- Unequal changes in the amplitude spectrum of a time-discrete signal are minimal when the sampler is disturbed by a static phase error. High-frequency components of a signal are shown to be somewhat more sensitive to a static phase error than low-frequency components. The only exception concerns frequency components of $f = \frac{1}{2T}$, where the distortion is significant.

E7 pre- and de-emphasis filtering is shown to give a maximum gain of 6 dB in signal-to-noise ratio at the output of the receiver. For E8 emphasis filtering the gain is at most 11 dB. This gain is obtained for high-frequency components of an incoming signal. The actual gain is very much dependent on the characteristics of the signal. In general, the gain of emphasis filtering will be diminished in real transmission situations. For E7-emphasis filtering the actual gain will be approximately 3 dB in signal-to-noise ratio; for E8-emphasis filtering this is about 6 dB. For all receiver configurations the noise power spectral density increases as the frequency increases.

Taking the human visual system into account noise at frequencies around 1.5 MHz is seen to be most disturbing. The gain of emphasis filtering is immediately perceptible for frequencies above 2 MHz. The obtained gains are the same as previously described.

The research carried out in this report is concerned with the HDTV satellite

transmission chain. The system parameters, however, are chosen to allow both satellite- and cable-transmission of the HD-MAC signal. Further research is therefore necessary to analyse the influence of noise and intersymbol interference in the cable-transmission chain. New elements that then have to be taken into account concern AM-VSB modulation and the special characteristics of the existing cable network. For this new research the analysis and the computer programs stated in this report are useful.

References.

1. Annegarn M.J.J.C., et al.,"HD–MAC: een stap vooruit in de evolutie van de televisietechniek". Philips Technisch Tijdschrift, vol. 43 (1986) nr.8, p.213–229
2. Wessels H., "Op weg naar High Definition Televisie". De Ingenieur, february 1989, p.9–14.
3. "High Definition Television transmission tests using a 12 GHz DBS channel HD–MAC system". CCIR, Study period 1986–1990, 10/11S–doc 199.
4. "Satellite transmission of multiplexed analogue component vision signals". CCIR, Study period 1986–1990, 10/11S–doc 154.
5. Shanmugam K.S., *Digital and analog communication systems.* Wiley, New York 1985.
6. "D2–HD–MAC/packet compatible HDTV transmission". CCIR, Study period 1986–1990, 10/11S–doc 131.
7. Mertens H. and D. Wood, "Standards proposed by the EBU for satellite broadcasting and cable distribution". J.IERE, vol. 56 (1986) no. 2, p. 53–61.
8. "The D–MAC/packet System for Satellite and Cable". IBA Technical Review no. 24 ,1988.
9. PC–Matlab User's Guide. The MathWorks, South Natick, Feb. 1989.
10. Kamen E., *Introduction to Signals and Systems.* Macmillan Publishing Company, New York 1987.

11. Kauff P., "Analysis of noise interferences in HDTV systems taking into account the properties of the human visual system". Picture Coding Symposium PCS'88, Torino Italy, Sept. 1988.

12. Kauff P. and R. Schäfer, "Multidimensional Noise Weighting in HDTV and its Applications to HD-MAC". Proceedings of the 3rd international workshop on HDTV, vol.1, Torino Italy, 30/31 Aug. & 1 Sept. 1989.

Appendix A: Practical implementations of digital half Nyquist filtering using transversal filters.

As stated in equation (2.2), the Nyquist filtering operation can be written as

$$y(kT') = \sum_{j=0}^{N-1} c_j \, x((k-j)T') \tag{A-1}$$

with notation $T' = \frac{T}{2}$. Now define

$$p_{\frac{1}{2}}'(kT') = \begin{cases} c_k & 0 \leq k < N \\ 0 & k \text{ elsewhere} \end{cases} \tag{A-2}$$

the impulse response of a nonideal digital half Nyquist filter. Then, (A–1) can be written as

$$y(kT') = (p_{\frac{1}{2}}' * x)(kT') \tag{A-3}$$

where * denotes a discrete convolution operation. From (A–2) and (A–3) it follows that the digital half Nyquist filter is totally described by the gains of the taps c_k and the sample frequency $\frac{1}{T}$. In the following, two half Nyquist filters are presented for a sample frequency of $\frac{1}{T'} = 40.5$ MHz. The filters are designed by use of the Remez optimal equirriple algorithm.

A.1. 20% roll-off, 15 taps Nyquist filter.

The gains of the taps are given in table A.1:

Table A.1: Gain coefficients of the 20% half Nyquist filter.

Coefficients	Gain
c_7	0.524684
$c_6 = c_8$	0.315317

$c_5 = c_9$	−0.025999
$c_4 = c_{10}$	−0.092148
$c_3 = c_{11}$	0.017081
$c_2 = c_{12}$	0.054002
$c_1 = c_{13}$	−0.031414
$c_0 = c_{14}$	−0.027208

Figure A.1 shows a drawing of the final shaping of the pulse. This is the impulse response of the full Nyquist filter, given by

$$p_1'(kT') = (p_{\frac{1}{2}}' * p_{\frac{1}{2}}')(kT')$$

(cascade of two half Nyquist filters).

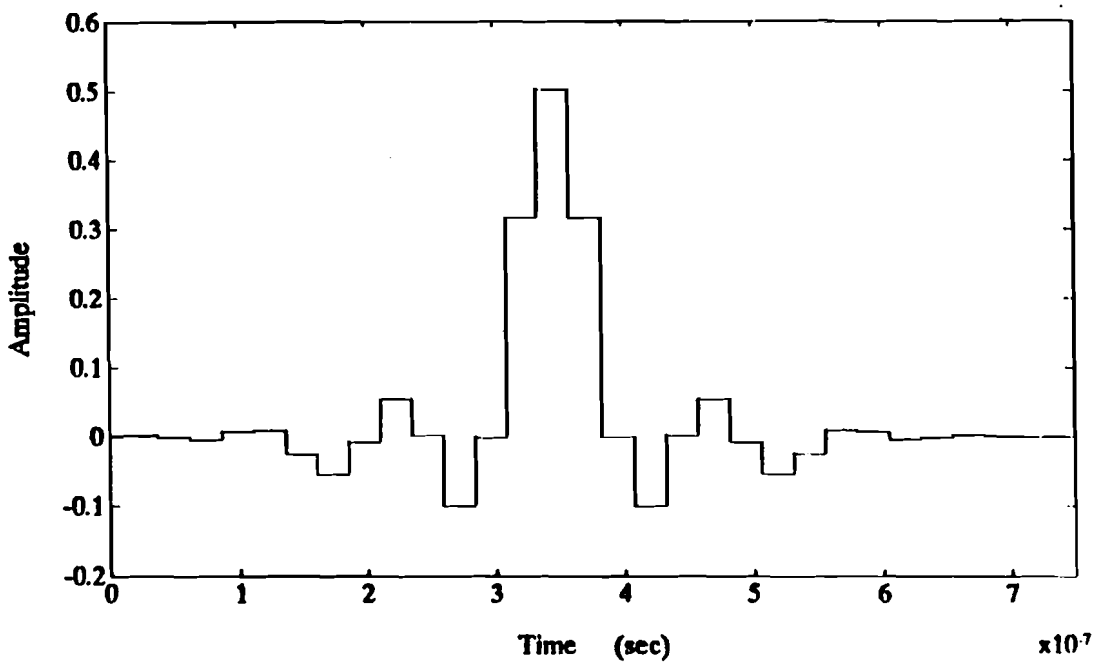

Figure A.1: Impulse response of the full digital Nyquist filter, 20% roll–off.

Figure A.2 shows the amplitude response of the 20% half Nyquist filter in the frequency domain, together with a drawing of the theoretical transfer function as given by equation (2.2). It can be seen that the digital filter is a good approximation of the theoretical optimum.

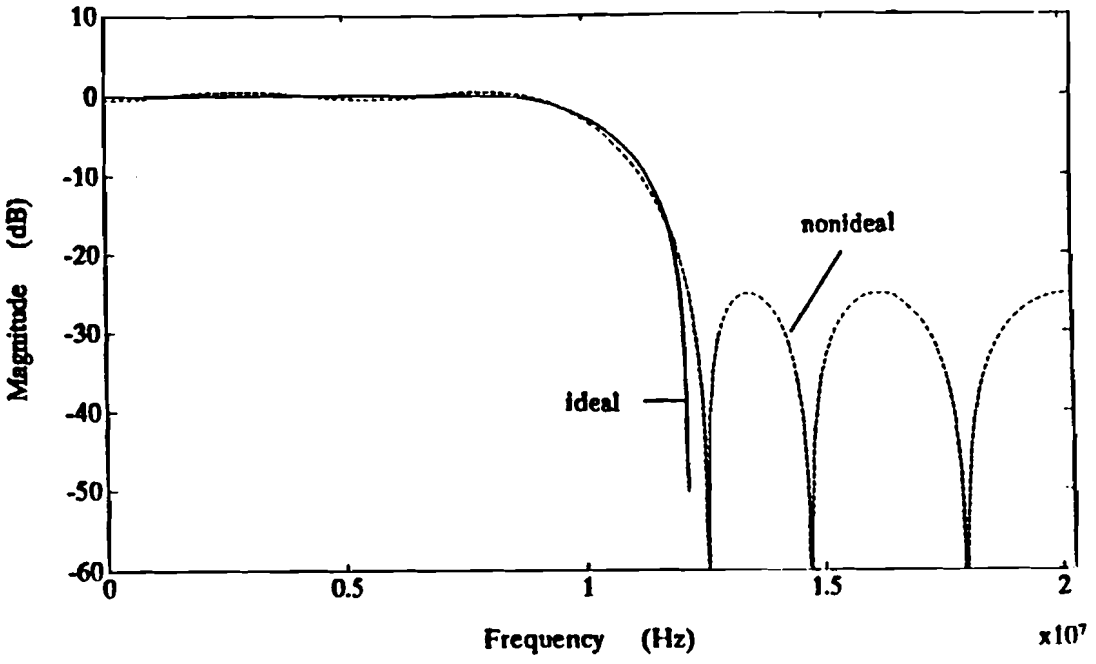

Figure A.2: Amplitude response of the 20% Nyquist filter; ideal and non–ideal.

A.2. 10% roll-off, 31 taps Nyquist filter.

The gains of the taps are given in table A.2. A drawing of the impulse response of the full Nyquist filter is given in figure A.3. Figure A.4 shows the amplitude response in the frequency domain, together with the theoretical transfer function (2.2) for $\alpha = 0.1$.

Table A.2: Gain coefficients of the 10% half Nyquist filter.

Coefficients	Gain
c_{15}	0.513569
$c_{14} = c_{16}$	0.317314
$c_{13} = c_{17}$	–0.013461
$c_{12} = c_{18}$	–0.103150

$c_{11} = c_{19}$	0.013014
$c_{10} = c_{20}$	0.058892
$c_9 = c_{21}$	–0.012478
$c_8 = c_{22}$	–0.038807
$c_7 = c_{23}$	0.011248
$c_6 = c_{24}$	0.027836
$c_5 = c_{25}$	–0.011532
$c_4 = c_{26}$	–0.017546
$c_3 = c_{27}$	0.005365
$c_2 = c_{28}$	0.021781
$c_1 = c_{29}$	–0.023173
$c_0 = c_{30}$	–0.016338

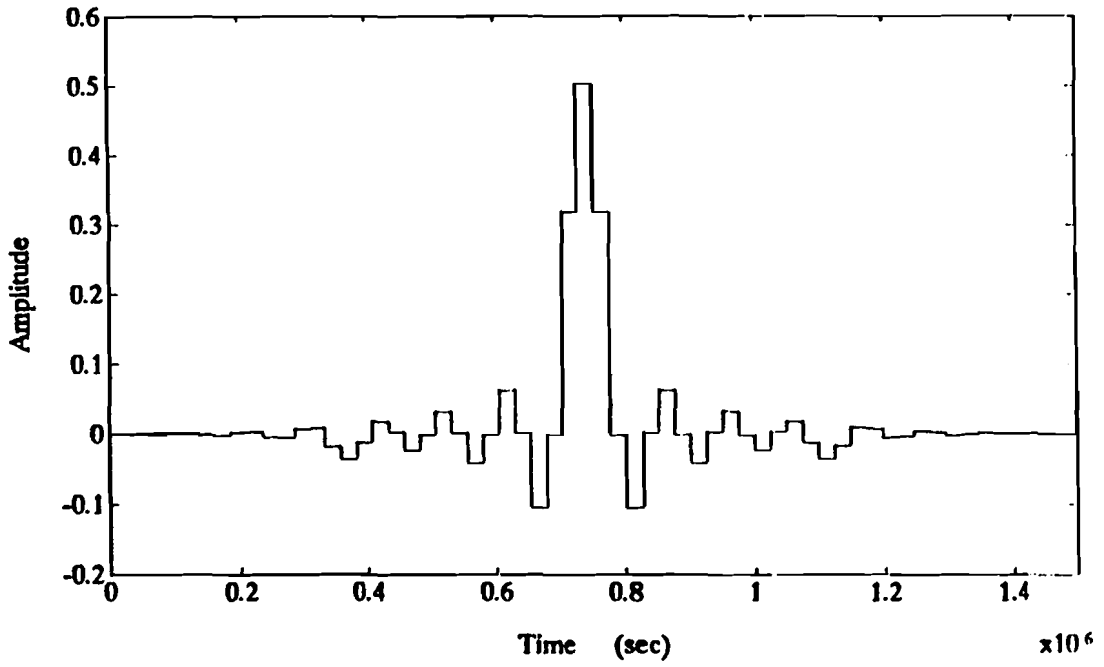

Figure A.3: Impulse response of the full digital Nyquist filter, 10% roll–off.

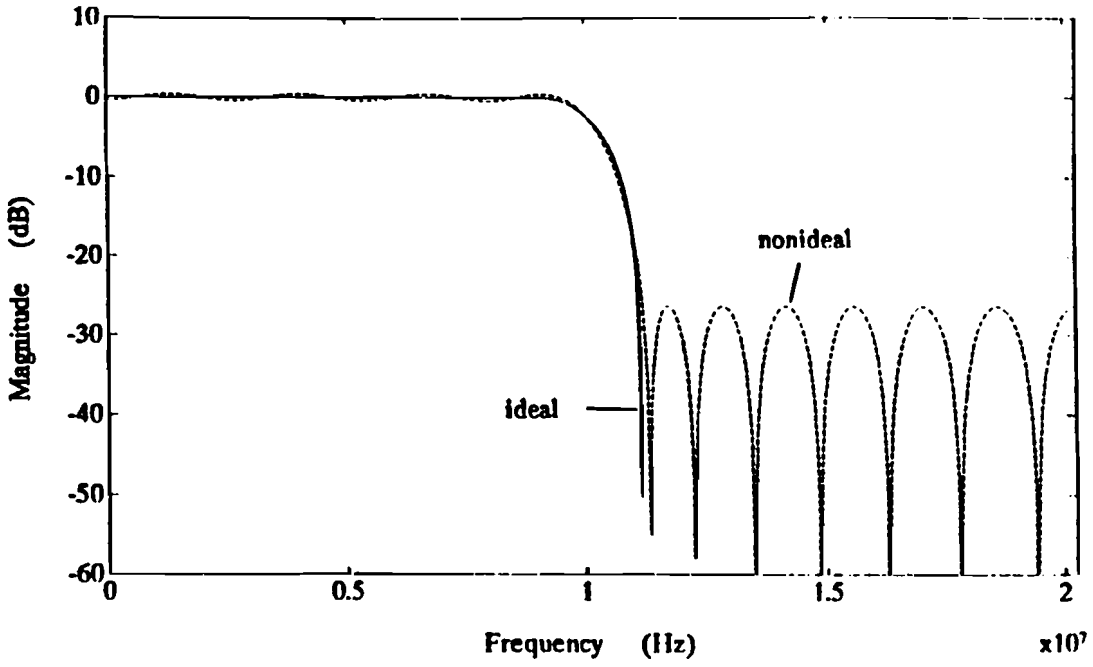

Figure A.4: Amplitude response of the 10% Nyquist filter; ideal and non–ideal.

Appendix B: Listings of the HD-MAC transmission chain MATLAB programs.

```
function y=e7preemp(x);
% function y=e7preemp(x);
%
%                    E7 nonlinear pre-emphase filter.
%
% Performs E7 nonlinear pre-emphase filtering. Note that the time vector is
% not changed. Time delay's will affect input vector x (by shifting right
% over 8 positions).
% E7PREEMP first looks for a file E7PRE.MAT which contains initial data of
% the delay's in the filter. For detailed information see the program text.
% E7PRE.MAT is automatically generated and saved in each function call, so
% for every next call the initial data will be available. A long file of
% input data x can therefore be cut into small vectors and subsequently
% filtered. The concatenation of the outputs y forms the filtering of x,
% without losing any information due to the cutting.
%
% See also: E7DEEMP, DATTOMAT, GEN_N7.PAS, GEN_N7INV.PAS.

% Define Gaussian Low Pass Filter with 9 taps :

F=[0.0029 0.0251 0.0972 0.2255 0.2986 0.2255 0.0972 0.0251 0.0029];

% Load initial data for the de-emphasis part if it exist.
% e7pre.mat contains (in order) :
%              prevx       (length 8)     previous signal x
%              prevy1      (length 4)     previous signal y1 (See sheet)
%              initF1      (length 8)     initial conditions filter        F1
%              initF2      (length 8)                                      F2.

if ( exist('e7pre.mat') >= 2 ),
     load e7pre.mat
     delete e7pre.mat                % Statement only works if e7pre.mat is
                                     % in the current directory.
else
     prevx=zeros(1,8);
     prevy1=zeros(1,4);
     initF1=zeros(1,8);
     initF2=zeros(1,8);
end;
prevxlow1=prevx(5:8);                % initial contents of delay      T1.
prevdelay1=prevx;                    %                                T3.
prevxlow2=prevy1;                    %                                T1(A).
prevdelay2=prevy1;                   %                                T2

% Check if n already exists as a variable in workspace. If not, load it:

if ( exist('n7') ~= 1 ),
     load n7.mat;
end
```

```
% Start filtering (up to input nonlinear filter N):

[xlow,endF1]=filter(F,1,x,initF1);
xd4=delay(x,4,prevxlow1);              % xd is the over 4 samples shifted version of x.
xhigh=xd4-xlow;

% Limit xhigh in the range [-0.5,0.5] :

limhigh=find(xhigh>0.5);
limlow=find(xhigh<-0.5);
xhsave=xhigh(limhigh);
xlsave=xhigh(limlow);
xhigh(limhigh)=0.5*ones(1,length(limhigh));
xhigh(limlow)=-0.5*ones(1,length(limlow));

% Continue :

y1=sign(xhigh).*n7([round(abs(xhigh)*4096+1)])/4096 + xlow;
y1(limhigh)=xhsave;
y1(limlow)=xlsave;
[y2,endF2,nextxlow2]=e7deemp(y1,initF2,prevxlow2);

xd8=delay(x,8,prevdelay1);
z1=y2-xd8;
y1d4=delay(y1,4,prevdelay2);
y=y1d4-z1;

% Save de-emphasis part initial data for the (eventual) next call :

lx=length(x);
prevx=x(lx-7:lx);
prevy1=y1(lx-3:lx);
initF1=endF1;
initF2=endF2;
save e7pre.mat prevx prevy1 initF1 initF2



function [x1,t1]=interpol(x0,t0);
% function [x1,t1]=interpol(x0,t0);
%
%                    Interpolate (x0,t0) with zeros
%
% INTERPOL performs the frequency doubling necessary before the Nyquist
% filtering can be carried out.
% x0=(a b c d e f .. )      becomes x1=(a 0 b 0 c 0 d 0 e 0 f 0 .. 0)
% t0=(ta tb tc .. )         becomes t1=(ta ta+T/2 tb tb+T/2 tc tc+T/2 ..)
% ( Here, tb-ta=T is assumed.)

T=1/20.25e6;
T2=T/2;
lx0=length(x0);
x1(1:2:2*lx0-1)=x0;
x1(2:2:2*lx0)=zeros(1,lx0);
t1(1:2:2*lx0-1)=t0;
t1(2:2:2*lx0)=t0+T2;
```

```
function [y,endxy]=nyquist(x,r,n,initxy)
% function [y,endxy]=nyquist(x,r,n,initxy)
%
%                    Nonideal digital half Nyquist filter.
%
% [y,endxy]=NYQUIST(x,r,n,initxy) with:
%    x          : input sequence
%    r          : roll off factor in percents, r=10% or r=20%
%    n          : number of taps used for filter
%    initxy     : initial conditions of delays (optional input, default zeros)
%    endxy      : final conditions of delays.
%
% TYPE NYQUIST for information on implemented filter choises.

if (nargin < 3 | nargin > 4),
      error('Incorrect number of input arguments.')
elseif (nargin == 3),
      initxy=zeros(1,n-1);
end

if (r==20 & n==15)
      coef=[     -.027208;-.031414; .054002; .017081;-.092148;-.025999;
                  .315317; .524684; .315317;-.025999;-.092148; .017081;
                  .054002;-.031414;-.027208];
      coef=coef';
elseif (r==20 & n==16)
      coef=[     -.005546;-.046722; .016297; .052913;-.039056;-.100173;
                  .128126; .466822; .466822; .128126;-.100173;-.039056;
                  .052913; .016297;-.046722;-.005546];
      coef=coef';
elseif (r==10 & n==31)
      coef=[     -.016338;-.023173; .021781; .005365;-.017546;-.011532;
                  .027836; .011248;-.038807;-.012478; .058892; .013014;
                 -.103150;-.013461; .317314; .513569];
      coef=coef';
      coef1=coef(15:-1:1);
      coef=[coef coef1];
elseif (r==10 & n==32)
      coef=[     -.002438;-.031221; .004540; .017090;-.005875;-.022276;
                  .010072; .029029;-.016746;-.038953; .028268; .056079;
                 -.052554;-.097769; .139424; .459418];
      coef=coef';
      coef1=coef(16:-1:1);
      coef=[coef coef1];
else
      disp(' No filter implemented with this roll off and number of taps')
      disp('')
      disp(' Possible at this moment : ')
      disp(' 20% roll off with 15 taps ; 20% roll off with 16 taps')
      disp(' 10% roll off with 31 taps ; 10% roll off with 32 taps')
      error(' Filter not available.')
end
[y,endxy]=filter(coef,1,x,initxy);
y=y/sum(coef);                          % Restore a gain of 1 .
```

```
function [y,ty,endxy]=dac_rec(x,tx,alfa,N,initxy);
% function [y,ty,endxy]=dac_rec(x,tx,alfa,N,initxy);
%
%          Digital-to-Analog Conversion and Reconstruction filter.
%
%           x,tx        : input signal (digital representation);
%           y,ty        : output signal  (analog representation);
%           N           : resolution of the output signal;
%           alfa        : roll off factor of the used Nyquist filter.
%
% tx must contain samples spaced at T/2 seconds.  ty represents an analog
% signal with resolution N: ty has its samples spaced at T/(2*N) seconds.
% DAC_REC(x,tx,alfa) assumes defaults N=10, initxy=zeros(1,..);
% DAC_REC(x,tx,alfa,N) assumes default initxy=zeros(1,..);

% Check input arguments :

if ( nargin == 3 ),
     N=10;
end

% Define constants :

T=1/20.25e6;
T2=T/2;
T3=T2/N;
lx=length(x);

% Interpolate (x,tx) with zeros :

xa=zeros(1:N*lx);
xa(1:N:N*lx)=x;
ty=tx(1):T3:tx(lx)+(N-1)*T3;

% Define ideal LPF :

th=-10*T:T3:10*T;
lh=length(th);
if ( lh > length(xa) ),
     error('Not enough input samples for correct filtering.');
end
h=sinc(th*(1+alfa)/T)*(1+alfa);

% Check other input arguments :

if ( nargin == 3 | nargin == 4 ),
     initxy=zeros(1,lh-1);
elseif ( length(initxy) == 0 ),
     initxy=zeros(1,lh-1);
end

% Filter :

[y,endxy]=filter(h,1,xa,initxy);                % Convolution.
```

```
function [y,endxy]=e1preemp(x,t,initxy);
% function [y,endxy]=e1preemp(x,t,initxy);
%
%                  E1 linear pre-emphasis filter.
%
%    x,t          : input signal;
%    initxy       : initial conditions of the delays (optional input);
%    y,t          : output signal;
%    endxy        : final conditions of the delays (to be used as initial conditions
%                   in the next call).
%
% The impuls response h of the E1 linear pre-emphasis filter is defined
% for th=0:Ts:5e-7 , with Ts the increase in time of the samples in vector
% t. Note that Ts must therefore be in the order of 1e-9. An error is given
% if LENGTH(h) > LENGTH(x). For Ts=1e-9, LENGTH(h)=501.
% E1PREEMP(x,t,-1) omits the time delay and gives a 'normal' CONV-output, but
% no account is taken to the initial and final states of the filter.

% N represents the resolution of t: Successive elements of t contain
% samples with a difference of Ts=T/(2*N).

T=1/20.25e6;
N=round(T/(2*(t(2)-t(1))));
if (N < 0), error('Time samples must be given in ascending order.'); end
if (N == 0), error('Time samples must have smaller increase in time.'); end
Ts=T/(2*N);

% Define filter constants :

A=1/sqrt(2);
f1=0.84e6;
f2=1.5e6;

% Compute impuls respons h of the E1 filter :

th=0:Ts:5e-7;
h=A*2*pi*f2*(1-f2/f1)*exp(-2*pi*f2*th);
% h(1)=h(1)+A*f2/f1;                    % This delta-part comes later.

% Check solvability :

lh=length(h);
lx=length(x);
if (lh > lx),
      error('Not enough input samples for correct filtering.');
end

% Instead of the function CONV.M , we use the build in filter-function, so
% initial conditions can be given.

if (nargin == 2),
      initxy=zeros(1,lh-1);
elseif (length(initxy) == 0),
      initxy=zeros(1,lh-1);
end
if (initxy == -1),
```

```
      [y,endxy]=filter(h,1,x);
      y=Ts*y+A*x*f2/f1;
else
      [y,endxy]=filter(h,1,x,initxy);
      y=Ts*y+A*x*f2/f1;                          % Delta-part added to y.
      endxy=Ts*endxy';
end



function y=sinc(x);
% function y=sinc(x);
%
%                Elementary sinc function
%
%                  y = sin(pi*x) / pi*x
% All abs(x)<1.-14 are considered zero (limit case).
%

z1=find(abs(x)<1.e-14);                  % x=0 is limit case,
x(z1)=ones(1,length(z1));                % so remove them.
y=sin(pi*x) ./ (pi*x);
y(z1)=ones(1,length(z1));                % limit value for x=0.



function [y,nextx]=delay(x,d,prevx);
% function [y,nextx]=delay(x,d,prevx);
%
%                      Delay function.
%
% Vector x is shifted d positions to the right.
% The information that comes shifted in  can be stored in prevx.
% The information that comes shifted out can be obtained from nextx.
% LENGTH(nextx)=LENGTH(prevx)=d.

if (d < 0),
      error('Delay times cannot be negative.');
end
if ( nargin == 2 ),
      prevx=zeros(1,d);
elseif ( nargin == 3 ),
      if (length(prevx) ~= d ),
            error('LENGTH(prevx) must be d.');
      end
else
      error('Incorrect number of input arguments.');
end

lx=length(x);
y=[ prevx x(1:lx-d) ];
nextx=x(lx-d+1:lx);
```

```
function [xc,endphase]=fmmod(x,t,fc,startphase);
% function [xc,endphase]=fmmod(x,t,fc,startphase);
%
%                          FM modulator.
%
%    x,t                 : input signal.
%    fc                  : modulation frequency.
%    startphase          : initial value of the phase (optional input, default 0).
%    xc,t                : FM modulated signal.
%    endphase            : end value of the phase.
%
% The frequency deviation constant is kf=13.5 MHz/V,
% the amplification is Ac=1.
%
% Note that vector t must meet the Nyquist criterium: If, for example,
% fc=70 MHz, the time samples must be separated by at most 1/(2*70e6) sec.
% (This is about 7e-9 sec !).
% endphase is meant to be used as startphase in an eventual next function
% call.

% Check input arguments:

if ( nargin < 3 | nargin > 4 ),
      error('Incorrect number of input arguments')
elseif ( nargin == 3 ),
      startphase=0;
end

Ts=t(2)-t(1);
if ( Ts < 0 ),
      error('Time samples must be in ascending order.');
elseif ( Ts > 1/(2*fc) ),
      error('Time samples must be seperated by at most 1/(2*fc).');
end

kf=13.5e6;
lx=length(x);

% The signal phase can be computed using the following statements:
%                       phase(1)=startphase;
%                       for k=2:lx, phase(k)=phase(k-1)+x(k-1)*Ts; end
% but it is done substantially faster using the build-in filter function:
[phase,endphase]=filter([0 1],[1 -1],x*Ts*kf,startphase);

phase=rem(phase,2*pi);
endphase=rem(endphase,2*pi);
xc=cos(2*pi*fc*t + phase);
```

```
function [y,endy,endsumy]=pll(xc,t,fc,inity,initsumy);
% function [y,endy,endsumy]=pll(xc,t,fc,inity,initsumy);
%
%                  FM demodulator (First order PLL).
%
%    xc,t              : input signal.
%    fc                : modulation frequency.
%    inity,initsumy    : initial values for y and the summation over y.
%                             (optional inputs, defaults zero).
%    y,t               : FM demodulated signal.
%    endy,endsumy      : final values of y and the summation over y.
%
% Note that vector t must meet the Nyquist criterium: If, for example,
% fc=70 MHz, the time samples must be seperated by at most 1/(2*70e6) sec.
% (This is about 7e-9 sec !).
% Maximum length of xc is 2504 (Due to the Turbo Pascal implementation).
% endy and endsumy are meant to be used in an eventual next function call
% as inity and initsumy. Otherwise, they have no relevance.
%
% See also : PLL.PAS

% Checking of input arguments :

if ( nargin < 3 | nargin > 5),
      error('Incorrect number of input arguments.');
end
Ts=t(2)-t(1);
if ( Ts < 0 ),
      error('Time samples must be in ascending order.');
elseif ( Ts > 1/(2*fc) ),
      error('Time samples must be seperated by at most 1/(2*fc).');
end
if (nargin == 3),
      inity=0;
      initsumy=0;
elseif (nargin ==4),
      initsumy=0;
end
lxc=length(xc);
if (lxc > 2504),
      error('Maximum length of xc that PLL can handle is 2504.');
end

% For speed considerations, the PLL implementation has been made
% in Turbo Pascal. We have to save the work-variables and call this routine
% called PLL. After demodulation, we fetch the result by loading Matlab-file
% PLLOUT.MAT .

disp(' Running PLL.EXE ... ');
save pllin xc t fc inity initsumy
!PLL
load pllout                              % Load vectors y and sumy.
disp(' Ready. ');

% y(1) contains inity. The relevant information is stored in y(2:lxc+1):
```

```
y=y(2:lxc+1);
endy=y(lxc);
endsumy=sumy;



function [y,endxy]=e1deemp(x,t,initxy);
% function [y,endxy]=e1deemp(x,t,initxy);
%
%                  E1 linear de-emphasis filter.
%
%    x,t          : input signal;
%    initxy       : initial conditions of the delays (optional input, default zeros);
%    y,t          : output signal;
%    endxy        : final conditions of the delays.
%
% The impuls response h of the E1 linear de-emphasis filter is defined
% for th=0:Ts:5e-7 , with Ts the increase in time of the samples in vector
% t. Note that Ts must therefore be in the order of 1e-9. An error is given
% if LENGTH(h) > LENGTH(x). For Ts=1e-9, LENGTH(h)=501.
% E1DEEMP(x,t,-1) omits the time delay and gives a 'normal' CONV-output, but
% no account is taken to the initial and final states of the filter.
%
% See also: E1PREEMP.

% Check the number of input arguments :

if ( nargin < 2 | nargin >3 ),
     error('Incorrect number of input arguments.');
end

% N represents the resolution of t: Successive elements of t contain
% samples with a difference of Ts=T/(2*N).

T=1/20.25e6;
N=round(T/(2*(t(2)-t(1))));
if (N  < 0), error('Time samples must be given in ascending order.'); end
if (N == 0), error('Time samples must have smaller increase in time.'); end
Ts=T/(2*N);

% Define filter constants :

A=sqrt(2);                              % Watch out !
f1=0.84e6;
f2=1.5e6;

% Compute impuls respons h of the (inverse) E1 filter :

th=0:Ts:5e-7;
h=A*2*pi*f1*(1-f1/f2)*exp(-2*pi*f1*th);
% h(1)=h(1)+A*f1/f2;                    % This part comes later.

% Check solvability :
```

```
lh=length(h);
lx=length(x);
if (lh > lx),
      error('Not enough input samples for correct filtering.');
end

% Instead of the function CONV.M , we use the build in filter-function, so
% initial conditions can be given.

if (nargin == 2 ),
      initxy=zeros(1,lh-1);
elseif (length(initxy) == 0),
      initxy=zeros(1,lh-1);
end

if (initxy == -1),
      [y,endxy]=filter(h,1,x);
      y=Ts*y+A*x*f1/f2;
else
      [y,endxy]=filter(h,1,x,initxy);
      y=Ts*y+A*x*f1/f2;
      endxy=Ts*endxy';
end



function [y,endxy]=lpf14(x,t,initxy);
% function [y,endxy]=lpf14(x,t,initxy);
%
%           Ideal Low Pass Filter with cut off frequency 14 Mhz.
%
%    x,t          : input signal;
%    initxy       : initial conditions of the delays (optional input, default zeros);
%    y,t          : output signal;
%    endxy        : final conditions of the delays (to be used as initial conditions
%                        in the next call).
%
% The impuls response h of the ideal LPF is given by a sinc function. This
% function is defined for th=-5e-7:Ts:5e-7 , with Ts the increase in time
% of the samples in vector t. Note that Ts must therefore be in the order
% of 1e-9. An error is given if LENGTH(h) > LENGTH(x). For Ts=1e-9,
% LENGTH(h)=1001.
% A time delay of 5e-7 is caused by the causality of the filter.
% LPF14(x,t,-1) omits the time delay and gives a 'normal' CONV-output, but
% no account is taken to the initial and final states of the filter.
%
% Check of input arguments :

if ( nargin < 2 | nargin > 3 ),
      error('Incorrect number of input arguments.');
end
Ts=t(2)-t(1);
if ( Ts < 0 ),
      error('Time samples must be in ascending order.');
end
```

```
% Define constants :

B=14e6;                                 % Cut off frequency B=14 MHz.
th=-5e-7:Ts:5e-7;
h=sinc(2*B*th)*2*B;                     % Impuls response of LPF.

lh=length(h);
lx=length(x);
if (lh > lx),
      error('Not enough input samples for correct filtering.');
end

% Instead of the function CONV.M , we use the build in filter-function, so
% initial conditions can be given.

if (nargin == 2 | length(initxy) == 0),
      initxy=zeros(1,lh-1);
end

if (initxy == -1),
      [y,endxy]=filter(h,1,x);
      endxy=endxy';                     % output endxy is a column vector.
      y=[y endxy]*Ts;
      y=y(501:lx+500);                  % th(501)=0, so from here.
else
      [y,endxy]=filter(h,1,x,initxy);
      y=y*Ts;                           % Scale the amplitude.
      endxy=Ts*endxy';
end
```

```
function [y,endxlow,nextx]=e7deemp(x,initxlow,prevx);
% function [y,endxlow,nextx]=e7deemp(x,initxlow,prevx);
%
%                       E7 de-emphase filter.
%
% Performs E7 non linear de-emphase filtering. Note that the time vector is
% not changed. Time delay's will affect input vector x (by shifting 4
% positions to the right).
% y=E7PREEMP(x) returns the filtered x in y.
% [y,endxlow,nextx]=E7PREEMP(x,initxlow,prevx) gives control over the initial
% and final states of the delays in the Gaussian F filter (As with the build
% in function FILTER). prevx must contain the last 4 samples of x of the
% previous e7 de-emphasis filtering to eliminate the effects of the delay.
%
% See also: E7DEEMP, DATTOMAT, GEN_N7.PAS, GEN_N7INV.PAS.

% Define Gaussian Low Pass Filter with 9 taps :

F=[0.0029 0.0251 0.0972 0.2255 0.2986 0.2255 0.0972 0.0251 0.0029];

% Check input arguments :
```

```
if ( nargin == 1 ),
      initxlow=zeros(1,8);
      prevx=[0 0 0 0];
elseif ( nargin == 3 ),
      if (length(prevx) ~= 4),
            error('Length of the prevx vector must be 4.');
      end
      if (length(initxlow) ~= 8),
            error('Length of the initxlow vector must be 8.');
      end
else
      error('Incorrect number of input arguments (must be 1 or 3).');
end

% Check if ninv already exists as a variable in workspace. If not, load it:

if ( exist('n7inv') ~= 1 ),
      load n7inv.mat;
end

% Start filtering:

[xlow,endxlow]=filter(F,1,x,initxlow);
[xd,nextx]=delay(x,4,prevx);              % xd is the over 4 samples shifted version of x
xhigh=xd-xlow;

% Limit xhigh in the range [-0.5,0.5], save overshoots :

limhigh=find(xhigh>0.5);
limlow=find(xhigh<-0.5);
xhsave=xhigh(limhigh);
xlsave=xhigh(limlow);
xhigh(limhigh)=0.5*ones(1,length(limhigh));
xhigh(limlow)=-0.5*ones(1,length(limlow));

% Continue :

y=sign(xhigh).*n7inv([round(abs(xhigh)*4096+1)])/4096 + xlow;

% Put back overshoots :

y(limhigh)=xhsave;
y(limlow)=xlsave;
```

Appendix C: Listings of the HD-MAC transmission chain PASCAL programs.

```
Program PLL (input,output);

{   Simulation of a Phase Locked Loop. Originally implemented as a M-file in
    Matlab. PLL loads a Matlab-file PLLIN.MAT , then simulates a demodulation
    using the PLL. After demodulation the result is saved as PLLOUT.MAT
    (also a Matlab-file).
    PLL constants :     VCO sensitivity constant    = 13.5e6 ;
                        Gain                        = 1      ;  }

{   Author of original Save and Load routines :     Charles D. Packard
                                                    The Mathworks, Inc.
    (They have been modified to serve the specific needs of PLL)    }

{$N+}                           { Switch numeric co-processor 8087 on }
{$M 65520,0,655360}             { Increase stack space                }

{   For a complete explanation of the format of a MATLAB data (.mat)
    file, see load and save in the MATLAB User's Guide Reference
    section.}

const
    kf          = 13.5e6;       { Frequency deviation constant }
    limity      = 10.0;         { Limit value for PLL output   }
    lbuf        = 2510;         { Length of buffer array's     }

type
    HeadRec   = record          {contents of standard 20 byte header}
        mattype : longint;      {describes machine that created mat file}
        matrows : longint;      {number of rows in data set}
        matcols : longint;      {number of columns in data set}
        imagef  : longint;      {1 if data set is complex, 0 if not}
        namelen : longint;      {number of characters in array name, +1 for
                                     the 0-byte used to end array string in C}
    end;
    Buffer    = array[1..lbuf] of double;
    NameStr   = array[1..10] of char;   {the name of the next variable in file}

var
    n                   : integer;      { Working variable }

                                        { Input variables to be read: }
    xc,t                : Buffer;           { modulated signal      }
    lxc                 : integer;          { length of xc          }
    fc                  : double;           { modulation frequency  }
    inity,initsumy      : double;           { initial values        }

                                        { VCO loop variables: }
    fvco                : double;           { vco frequency         }
    c0,c1,c2            : double;           { hunting variables     }
    ynew,ph,delta       : double;
```

```
eta             : real;                 { hunting accuracy          }
sgn             : boolean;              { sign memory               }
numiter         : integer;              { number of iterations      }

                                      { PLL variables: }
y               : Buffer;               { output of PLL             }
sumy            : double;               { summation over y          }
wc              : real;                 { wc=2*pi*fc                }
Ts              : double;               { sampling time             }
xcn             : double;               { xcn=xc[n]                 }


procedure loadmat(  var xc              : Buffer;
                    var lxc             : integer;
                    var t               : Buffer;
                    var fc              : double;
                    var inity,initsumy  : double);

{   Load variables needed for demodulation. Loadmat searches for the
    Matlab-file PLLIN.MAT which must contain the following variables
    (in order): xc t fc inity initsumy                              }

const
    nvar=5;             { number of variables to be read            }
    nsingle=3;          { number of doubles among these variables   }

var
    in_file     : file;
    ZeroByte    : byte;
    header      : array[1..nvar] of HeadRec;
    name        : array[1..nvar] of NameStr;
    bufp        : array[1..nvar-nsingle] of Buffer;
    doubp       : array[1..nsingle] of double;
    i           : integer;

begin
    assign(in_file,'pllin.mat');
    reset(in_file, 1);
    for i:=1 to nvar-nsingle do
    begin
        blockread(in_file, header[i], 20);
        blockread(in_file, name, header[i].namelen - 1);
        blockread(in_file, ZeroByte, 1);
        blockread(in_file, bufp[i], 8*header[i].matcols*header[i].matrows);
    end;
    for i:=nvar-nsingle+1 to nvar do
    begin
        blockread(in_file, header[i], 20);
        blockread(in_file, name, header[i].namelen - 1);
        blockread(in_file, ZeroByte, 1);
        blockread(in_file, doubp[i-nsingle+1], 8);
    end;
    xc:=bufp[1];
    lxc:=header[1].matcols;
    t:=bufp[2];
    fc:=doubp[1];
```

```
        inity:=doubp[2];
        initsumy:=doubp[3];
        close(in_file);
    end;


    procedure savemat(y : Buffer;sumy : double);
    { Save demodulated output y as Matlab-file PLLOUT.MAT }

    const
        ZeroByte   : byte = 0; {NUL terminator in a C character string}

    var
        out_file   : file;
        header     : HeadRec;
        name       : NameStr;

    begin
        assign(out_file, 'pllout.mat');
        rewrite(out_file, 1);
        with header do
        begin
            mattype:=0;
            matrows:=1;
            matcols:=lxc+1;
            imagef :=0;
            namelen:=2;
        end;
        name[1]:='y';
        blockwrite(out_file, header, 20);
        blockwrite(out_file, name, header.namelen-1);
        blockwrite(out_file, ZeroByte, 1);
        blockwrite(out_file, y, 8*header.matcols*header.matrows);
        with header do
        begin
            mattype:=0;
            matrows:=1;
            matcols:=1;
            imagef :=0;
            namelen:=5;
        end;
        name[1]:='s';name[2]:='u';name[3]:='m';name[4]:='y';
        blockwrite(out_file, header, 20);
        blockwrite(out_file, name, header.namelen-1);
        blockwrite(out_file, ZeroByte, 1);
        blockwrite(out_file, sumy, 8);
        close(out_file);
    end;

begin
    loadmat(xc,lxc,t,fc,inity,initsumy);
    wc:=2*pi*fc;
    Ts:=t[2]-t[1];
    sumy:=initsumy;
```

```
    y[1]:=inity;
    eta:=1e-9;
    for n:=1 to lxc do
    begin
        xcn:=xc[n];
        ynew:=y[n];
        sumy:=sumy+ynew;
        fvco:=wc*t[n]+sumy*kf*Ts;
        ph:=0;
        numiter:=0;
        delta:=0.1;
        c0:=cos(fvco)-xcn;
        while (abs(c0) > eta) and (numiter < 50) do
        begin
            sgn:=(c0 >= 0);
            c1:=cos(fvco+kf*Ts*(ph+delta));
            c2:=cos(fvco+kf*Ts*(ph-delta));
            if (abs(c1-xcn) > abs(c2-xcn))      then ph:=ph-delta
                                                else ph:=ph+delta;
            c0:=cos(fvco+ph*kf*Ts)-xcn;
            if ( (c0 >= 0) <> sgn ) then delta:=delta/2;
            inc(numiter);
        end;
        if ( ynew+ph >  limity ) then ph:= limity-ynew;
        if ( ynew+ph < -limity ) then ph:=-limity-ynew;
        y[n+1]:=ynew+ph;
    end;
    sumy:=sumy+ph;

{   To obtain the PLL output without high frequency components, y must be
    processed by a Low Pass Filter with a cut off frequency of about 20 MHz. }

    savemat(y,sumy);                { Save results in a MAT-file }

end.
```

```
program GEN_N7 (input,output) ;

{ Original program is written by        L. Vervoort
                                        Consumer Electronics
                                        Philips Eindhoven
  The original program has been modified to serve the specific needs of
  the MATLAB implementations.                                          }

{$N+}

type
     lookup = array[0..2048] of integer ;

var
     lookupfile                         : text ;
     n7                                 : lookup ;
     i                                  : integer ;

procedure find_n7(var n : lookup) ;

     const
          a = 0.011 ;
          b = 19.80 ;
          c = 1.5225 ;

     var
          i                  : integer ;
          ac,ac2,acln        : real ;
          vi,vo,vt           : real ;

     begin
          ac  := 2 * a * c ;
          ac2 := ac * ac ;
          acln := ln(ac) ;
          for i := 0 to 2048 do
          begin
               vi := int(i)/int(4096) ;
               vt := vi + sqrt(vi * vi + ac2) ;
               vo := vi/c + ( ln(vt) - acln ) / b ;
               n7[i] := round( 4096 * vo ) ;
          end ;
     end ;

begin
     writeln;
     writeln(' Deriving non linear function N7 ... ');
     find_n7(n7) ;

     { save N to disk }
     writeln(' Saving N7 as ASCII file under name N7.DAT ... ');
     assign(lookupfile,'n7.dat') ;
     rewrite(lookupfile) ;
     for i:=0 to 2048 do write(lookupfile,n7[i],' ') ;
     close(lookupfile) ;
end.
```

```
program GEN_N7INV (input,output) ;

{ Original program is written by        L. Vervoort
                                        Consumer Electronics
                                        Philips Eindhoven
  The original program has been modified to serve the specific needs of
  the MATLAB implementations.                                          }

{$R+}
{$N+}
{$M 65520,0,655360}

type
     lookup = array[0..2048] of integer ;

var
     lookupfile                         : text ;
     n7_inv                             : lookup ;
     i                                  : integer ;

procedure find_n7_inv(var n_inv : lookup) ;

     const
          a = 0.011 ;
          b = 19.80 ;
          c = 1.5225 ;

     var
          i                  : integer ;
          ac,ac2,acln        : real ;
          vi,vo,vt           : real ;
          nt                 : array[0..2048] of integer ;
          vot                : array[0..2048] of real ;

     begin
          ac   := 2 * a * c ;
          ac2  := ac * ac ;
          acln := ln(ac) ;
          for i := 0 to 2048 do
          begin
               n7_inv[i] := -10 ;
               vi := int(i)/int(4096) ;
               vt := vi + sqrt(vi * vi + ac2) ;
               vo := (vi/c + ( ln(vt) - acln ) / b) * int(4096) ;
               vot[i] := vo ;
               nt[i] := round( vo ) ;
          end ;

          for i := 0 to 2048 do
          begin
               if n7_inv[nt[i]] = -10
               then
               begin
                    n7_inv[nt[i]] := i ;
               end
               else
```

```
            begin
                if abs(vot[i] - nt[i]) < abs(vot[i-1] - nt[i])
                then n7_inv[nt[i]] := i ;
            end ;
        end ;

        for i := 0 to 2048 do
        begin
            if n7_inv[i] = -10 then
                n7_inv[i] :=
                n7_inv[i-1] + round((int(nt[i])-vot[i])/(vot[i+1]-vot[i])) ;
        end ;

    end ;

begin

    writeln;
    writeln(' Deriving inverse of non linear function N7 ... ');

    find_n7_inv(n7_inv) ;

    { save N7_INV to disk }
    writeln(' Saving inverse N7 as ASCII file under name N7INV.DAT ... ');
    assign(lookupfile,'n7inv.dat') ;
    rewrite(lookupfile) ;
    for i:=0 to 2048 do write(lookupfile,n7_inv[i],' ') ;
    close(lookupfile) ;

end.
```

Appendix D: Listings of the HD-MAC transmission chain additional MATLAB programs

```
echo on
clear
%
%                    Mother file.
%
%    Simulates a HDTV satellite transmission system.

% Define constants :

fname=input('Give results filename : ','s');
T=1/20.25e6;
alfa=0.1;r=10;n=31;
N=10;
fc=70e6;

% Make 'vision' data files :

t0=0:T:99*T;
x0=[ zeros(1,9) ones(1,91)*0.3 ];
eval(['save ',fname,'01 x0 t0']);

% E7 pre-emphasis :

delete e7pre.mat
i=1;
while ( eval(['exist(''',fname,'0',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'0',int2str(i)]);
      x1=e7preemp(x0);
      t1=t0;
      eval(['save ',fname,'1',int2str(i),' x1 t1']);
      i=i+1;
end

% Interpolate with zeros ;

i=1;
while ( eval(['exist(''',fname,'1',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'1',int2str(i)]);
      [x2,t2]=interpol(x1,t1);
      eval(['save ',fname,'2',int2str(i),' x2 t2']);
      i=i+1;
end

% 1/2 Nyquist; Nonideal filter roll off 20%, 15 taps :

i=1;
initxy=zeros(1,n-1);
while ( eval(['exist(''',fname,'2',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'2',int2str(i)]);
```

```
      [x3,endxy]=nyquist(x2,r,n,initxy);
      t3=t2;
      initxy=endxy;
      eval(['save ',fname,'3',int2str(i),' x3 t3']);
      i=i+1;
end
clear t0 x0 t1 x1 t2 x2 t3 x3

% Low Pass filtering with cut off frequency 10.125+(roll off) :
% ( equals DAC + reconstruction filter )

i=1;
initxy=[];
while ( eval(['exist(''',fname,'3',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'3',int2str(i)]);
      [x4,t4,endxy]=dac_rec(x3,t3,alfa,N,initxy);
      initxy=endxy;
      eval(['save ',fname,'4',int2str(i),' x4 t4']);
      i=i+1;
end
clear t4 x4

% E1 linear pre-emphasis filtering :

pack
i=1;
initxy=[];
while ( eval(['exist(''',fname,'4',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'4',int2str(i)]);
      [x5,endxy]=e1preemp(x4,t4,initxy);
      initxy=endxy;
      t5=t4;
      eval(['save ',fname,'5',int2str(i),' x5 t5']);
      i=i+1;
end
clear t4 x4

% FM modulator :

i=1;
startphase=0;
while ( eval(['exist(''',fname,'5',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'5',int2str(i)]);
      [x6,endphase]=fmmod(x5,t5,fc,startphase);
      t6=t5;
      eval(['save ',fname,'6',int2str(i),' x6 t6']);
      i=i+1;
end
clear t5 x5

% FM demodulation using the PLL :

pack;
i=1;
inity=0;initsumy=0;
while ( eval(['exist(''',fname,'6',int2str(i),'.mat'')']) >= 2 ),
```

```
      eval(['load ',fname,'6',int2str(i)]);
      [y7,endy,endsumy]=pll(x6,t6,fc,inity,initsumy);
      inity=endy;
      initsumy=endsumy;
      t7=t6;
      eval(['save ',fname,'7',int2str(i),' y7 t7']);
      i=i+1;
end
clear y6 t6

% Low Pass Filter with cut off frequency B=14 MHz (Band limiter) :

i=1;
initxy=[];
while ( eval(['exist(''',fname,'7',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'7',int2str(i)]);
      [y8,endxy]=lpf14(y7,t7,initxy);
      endxy=initxy;
      t8=t7;
      eval(['save ',fname,'8',int2str(i),' y8 t8']);
      i=i+1;
end
clear t7 y7

% E1 de-emphasis filter :

i=1;
initxy=[];
while ( eval(['exist(''',fname,'8',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'8',int2str(i)]);
      [y9,endxy]=e1deemp(y8,t8,initxy);
      endxy=initxy;
      t9=t8;
      eval(['save ',fname,'9',int2str(i),' y9 t9']);
      i=i+1;
end
clear t8 y8

% Sampler :

if ( eval(['exist(''',fname,'91.mat'')']) >= 2 ),
      eval(['load ',fname,'91']);
      pos=find(abs(t9)<1e-12);
      pos=rem(pos(1),N);
      if (pos == 0), pos=N; end;
      y10=y9(pos:N:length(y9));
      t10=t9(pos:N:length(t9));
      eval(['save ',fname,'101 y10 t10']);
end
i=2;
while ( eval(['exist(''',fname,'9',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'9',int2str(i)]);
      y10=y10(pos:N:length(y9));
      t10=t10(pos:N:length(t9));
      eval(['save ',fname,'10',int2str(i),' y10 t10']);
```

```
      i=i+1;
end
clear pos t9 y9

% Half receiver Nyquist filter :

i=1;
initxy=zeros(1,n-1);
while ( eval(['exist(''',fname,'10',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'10',int2str(i)]);
      [y11,endxy]=nyquist(y10,r,n,initxy);
      t11=t10;
      initxy=endxy;
      eval(['save ',fname,'11',int2str(i),' y10 t10']);
      i=i+1;
end

% Subsampler :

i=1;
while ( eval(['exist(''',fname,'11',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'11',int2str(i)]);
      y12=y11(1:2:length(y11));
      t12=t11(1:2:length(t11));
      eval(['save ',fname,'12',int2str(i),' y12 t12']);
      i=i+1;
end

% E7 de-emphasis :

i=1;
initxlow=zeros(1,8);
prevx=zeros(1,4);
while ( eval(['exist(''',fname,'12',int2str(i),'.mat'')']) >= 2 ),
      eval(['load ',fname,'12',int2str(i)]);
      [y13,endxlow,nextx]=e7deemp(y12,initxlow,prevx);
      initxlow=endxlow;
      prevx=nextx;
      t13=t12;
      eval(['save ',fname,'13',int2str(i),' y13 t13']);
      i=i+1;
end
```

```
function dattomat
% function dattomat
%
%                   Convert .DAT files to .MAT files.

%          1. Load files N7.DAT, N7INV.DAT, N8.DAT, N8INV.DAT;
%          2. Save them as MAT variables;
%          3. Clear N7, N8 and N7INV, N8INV from workspace.
%

load n7.dat;
save n7.mat n7;
load n7inv.dat;
save n7inv.mat n7inv;

load n8.dat;
save n8.mat n8;
load n8inv.dat;
save n8inv.mat n8inv;

clear n7 n7inv n8 n8inv;




%
% Maximal ISI caused by a random phase error in the sampling clock.
%          Simulation Program for ideal Nyquist filtering.
%

T=1/20.25e6;
alfa=0.1;                              % Nyquist filter roll-off factor.
maxiter=1000;                          % Number of iterations.

% Determine output as function of the white jitter.
% Theoretical lower bound (ideal nyquist filtering) :

t=-30*T:T:30*T;
rand('normal');
for m=0:10,
      m
      sigma=m*1e-9;
      ISItot=0;
      for k=1:maxiter;
            noise=abs(rand(1,1))*sigma;
            part1=sum(abs(raisdcos(t+noise,alfa)))-abs(raisdcos(noise,alfa));
            ISItot=ISItot+part1*100/abs(raisdcos(noise,alfa));
      end
      ISI(m+1)=ISItot/maxiter;
end

sigma=0:1e-9:10e-9;
ISI10i=ISI;                            % The result is saved as ISI10i:
save ISI10i sigma ISI10i               % (10% roll-off, ideal Nyquist filters).
```

```
%
% Maximal ISI caused by a random phase error in the sampling clock.
%         Simulation program for nonideal Nyquist filtering.
%

T=1/20.25e6;
T2=T/2;
N=80;
r=10;n=31;alfa=0.1;                    % Nyquist filter parameters.
centre=55;

% input signal (delta puls) :

x0=zeros(1,101);x0(5)=1;
t0=0:T/2:50*T;

% first half nyquist filtering :

x1=nyquist(x0,r,n);
t1=t0;
lx1=length(x1);

% Low Pass Filter :

[x2,t2]=dac_rec(x1,t1,alfa,N);
lx2=length(x2);

% Find the right sample position :

pos=find(abs(t2)<1e-12);
pos=rem(pos(1),N);
if (pos==0), pos=N; end;

% Determine output as function of the white jitter :

rand('normal');
for m=0:10,
      sigma=m*1e-9;
      ISItot=0;
      for k=1:1000;
            noise=abs(rand(1,1))*sigma;
            shift=round(noise*N/T2);
            posnew=pos+shift;
            x3=x2(posnew:N:lx2);
            x4=nyquist(x3,r,n);
            x4=x4/x4(centre);                    % Normalize to 1.
            x4=x4(1:2:length(x4));
            ISItot=ISItot+(sum(abs(x4))-1)*100;
      end
      ISI(m+1)=ISItot/1000;
end

sigma=0:1e-9:10e-9;
ISI10=ISI;
save ISI10 sigma ISI10
```

```
%
% Maximal ISI caused by a static phase error in the sampling clock.
%   Simulation Program for ideal- and nonideal Nyquist filtering.
%

T=1/20.25e6;
T2=T/2;
N=20;
r=10;n=31;alfa=0.1;                    % Nyquist filter parameters.
if (n==31),
      centre=55;
elseif (n==15),
      centre=39;
else
      error(' Not implemented for this number of tabs.');
end

% input signal (delta puls) :

x0=zeros(1,101);x0(5)=1;
t0=0:T/2:50*T;

% first half nyquist filtering :

x1=nyquist(x0,r,n);
t1=t0;
lx1=length(x1);

% Low Pass Filter :

[x2,t2]=dac_rec(x1,t1,alfa,N);
lx2=length(x2);

% Find the right sample position :

pos=find(abs(t2)<1e-12);
pos=rem(pos(1),N);
if (pos==0), pos=N; end;

% Determine output as function of the static jitter :

ISI=zeros(1,6);
for spe=0:10,
      posnew=pos+spe;
      x3=x2(posnew:N:lx2);
      x4=nyquist(x3,r,n);
      x4=x4/x4(centre);                % Normalize to 1.
      x4=x4(1:2:length(x4));
      ISI(spe+1)=(sum(abs(x4))-1)*100;
end

% Theoretical lower bound (ideal nyquist filtering) :

ISIt=zeros(1,6);
t=-30*T:T:30*T;
for k=0:10,
```

```
      spe=k*T2/N;
      part1=sum(abs(raisdcos(t+spe,alfa)))-abs(raisdcos(spe,alfa));
      ISIt(k+1)=part1*100/abs(raisdcos(spe,alfa));
end;

% Save results :

spe=0:10;spe=spe*T2/N;
stat10=ISI;stat10t=ISIt;
save stat10 spe stat10 stat10t




%
% Noise power spectral density at the output of the receiver.
%      Simulation Program for nonideal Nyquist filtering.
%

cndb=18;                                % Carrier to Noise ratio (in dB).
ln=512;                                 % length of noise sample array.
alfa=0.1;r=10;ntap=31;                  % Nyquist filter constants.
rand('normal');
sigma=sqrt(0.04517/(10.^(cndb/10)));

% Read number of iterations :

numit=input('Give number of iterations : ');

% noise filter (PLL & E1 & sampler)

F1=[0.876254 0 -3.37541e-1 0 7.58777e-3 0 -4.25219e-2 0 1.51246e-3 ..
      0 -1.69417e-2 0];
F2=F1(11:-1:2);
F=[F2 F1];

% First run : Simulate for no de-emphasis filtering.

N=zeros(1,ln);
f=(0:ln-1)/(ln*T);
for num1=1:numit,
      n=rand(1,2*ln)*sigma;
      noise1=filter(F,1,n);                 % noise at output ADC.
      noise2=nyquist(noise1,r,ntap);        % noise at output Nyquist filter.
      noise3=noise2(1:2:2*ln);              % noise after subsampling.
      N3=fft(noise3);
      N=N+N3.*conj(N3)/ln;
end
N=N/numit;

% Second run : Simulate with E7 de-emphasis filtering.

Ne7=zeros(1,ln);
f=(0:ln-1)/(ln*T);
```

```
for num2=1:numit,
      n=rand(1,2*ln)*sigma;
      noise1=filter(F,1,n);                  % noise at output ADC.
      noise2=nyquist(noise1,r,ntap);         % noise at output Nyquist filter.
      noise3=noise2(1:2:2*ln);               % noise after subsampling.
      noise4=e7deemp(noise3);                % noise after E7 de-emphasis.
      N4=fft(noise4);
      Ne7=Ne7+N4.*conj(N4)/ln;
end
Ne7=Ne7/numit;

% Third run : Simulate with E8 de-emphasis filtering.

Ne8=zeros(1,ln);
f=(0:ln-1)/(ln*T);
for num3=1:numit,
      n=rand(1,2*ln)*sigma;
      noise1=filter(F,1,n);                  % noise at output ADC.
      noise2=nyquist(noise1,r,ntap);         % noise at output Nyquist filter.
      noise3=noise2(1:2:2*ln);               % noise after subsampling.
      noise4=e8deemp(noise3);                % noise after E8 de-emphasis.
      N4=fft(noise4);
      Ne8=Ne8+N4.*conj(N4)/ln;
end
Ne8=Ne8/numit;

% Save variables :

save npsd f N Ne7 Ne8 cndb numit alfa ln
```

Appendix E: Derivation of equation (6.7)

In this appendix the amplitude spectrum of a discrete signal of the form

$$z(kT) = \cos(2\pi f_b kT + 2\pi f_b \Delta T)$$

is derived. The FFT is defined as

$$Z(n) = \sum_{k=0}^{N-1} z(kT).\exp(j\frac{2\pi}{N}kn) \qquad n = 0,1,..,N-1$$

The derivation is as follows:

$$\begin{aligned} z(kT) &= \cos(2\pi f_b kT + 2\pi f_b \Delta T) \\ &= \frac{1}{2}\left\{ e^{j\omega(kT+\Delta T)} + e^{-j\omega(kT+\Delta T)} \right\} \end{aligned}$$

where $\omega = 2\pi f_b$.

With $\alpha = 2\pi(f_b T - \frac{n}{N})$

and $\beta = 2\pi(f_b T + \frac{n}{N})$

the FFT of $z(kT)$ can be written as

$$Z(n) = \frac{1}{2}\sum_{k=0}^{N-1} \left\{ e^{j(\omega\Delta T + \alpha k)} + e^{-j(\omega\Delta T + \beta k)} \right\}$$

Using the standard series

$$\sum_{k=0}^{N-1} x^k = \frac{1-x^N}{1-x}$$

we have

$$\sum_{k=0}^{N-1} e^{jk\alpha} = \frac{\sin(\frac{\alpha N}{2})}{\sin(\frac{\alpha}{2})}.e^{j\alpha\frac{N-1}{2}}$$

Let $A(n) = \dfrac{\sin(\frac{\alpha N}{2})}{\sin(\frac{\alpha}{2})}$

$$B(n) = \frac{\sin(\frac{\beta N}{2})}{\sin(\frac{\beta}{2})}$$

we then have

$$Z(n) = \frac{1}{2}\left\{ A(n).e^{j(\omega\Delta T + \alpha\frac{N-1}{2})} + B(n).e^{-j(\omega\Delta T + \beta\frac{N-1}{2})} \right\}$$

Furthermore,

$$|Z(n)|^2 = Z(n).Z^*(n)$$

or

$$|Z(n)|^2 = \frac{1}{4}\left\{ A(n)^2 + B(n)^2 + 2A(n)B(n)\cos\left[(\alpha+\beta)\frac{N-1}{2} + 2\omega\Delta T\right] \right\}$$

Appendix F: Specifications of the Noise filter.

The noise filter is a filter that transforms noise with a Gaussian distribution into noise with a distribution at the output of the sampler. It is an approximation of noise samples as processed in the PLL, the E1 de-emphasis, the LPF and the ADC (sampler). It is a digital transversal filter with 21 taps at a sample frequency of 40.5 MHz. The gain coefficients of the taps are given in table F.1.

Table F.1: Gain coefficients of the Noise filter.

Coefficients	Gain
$c_0 = c_{21}$	-1.69417e-2
$c_1 = c_{20}$	0
$c_2 = c_{19}$	1.51246e-3
$c_3 = c_{18}$	0
$c_4 = c_{17}$	-4.25219e-2
$c_5 = c_{16}$	0
$c_6 = c_{15}$	7.58777e-3
$c_7 = c_{14}$	0
$c_8 = c_{13}$	-3.37541e-1
$c_9 = c_{12}$	0
$c_{10} = c_{11}$	0.876254

Appendix G: Specifications of the Video Weighting filter.

The video weighting filter is a filter that approximates the sensitivity of the human visual system to various frequencies. It is a digital transversal filter with 41 taps at a sample frequency of 20.25 MHz. The gain coefficients of the taps are given in table G.1, the amplitude spectrum of the impulse response is given in figure G.1.

Table G.1: Gain coefficients of the Video Weighting filter.

Coefficients	Gain
$c_0 = c_{40}$	0.116939866e–2
$c_1 = c_{39}$	0.644675456e–3
$c_2 = c_{38}$	0.716075767e–3
$c_3 = c_{37}$	0.103168073e–2
$c_4 = c_{36}$	0.121486606e–2
$c_5 = c_{35}$	0.168500724e–2
$c_6 = c_{34}$	0.205384195e–2
$c_7 = c_{33}$	0.279401615e–2
$c_8 = c_{32}$	0.348338112e–2
$c_9 = c_{31}$	0.470168516e–2
$c_{10} = c_{30}$	0.596279278e–2
$c_{11} = c_{29}$	0.804857910e–2
$c_{12} = c_{28}$	0.103646815e–1
$c_{13} = c_{27}$	0.141008161e–1
$c_{14} = c_{26}$	0.184768550e–1
$c_{15} = c_{25}$	0.256257392e–1
$c_{16} = c_{24}$	0.344462395e–1
$c_{17} = c_{23}$	0.500149131e–1
$c_{18} = c_{22}$	0.708963275e–1
$c_{19} = c_{21}$	0.122608006
c_{20}	0.238043785

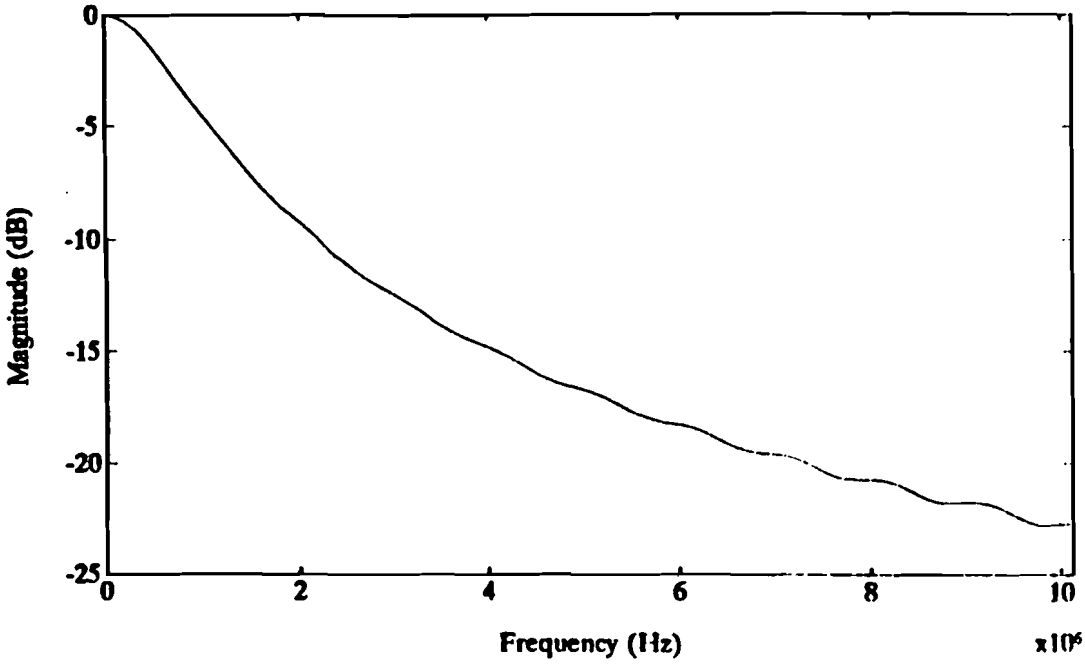

Figure G.1: Amplitude spectrum of the Video Weighting filter.