Discrete-Time Randomized Sampling

by

Maya Rida Said

B.S. Biology
B.S. Electrical Engineering and Computer Science
Massachusetts Institute of Technology, 1998
M.S. Toxicology
Massachusetts Institute of Technology, 2000

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2001

Author ....
Department of Electrical Engineering and Computer Science
December 22, 2000

Certified by ....
....
Alan V. Oppenheim
Ford Professor of Electrical Engineering
Thesis Supervisor

Accepted by ....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

Discrete-Time Randomized Sampling

by

Maya Rida Said

Submitted to the Department of Electrical Engineering and Computer Science
on December 22, 2000, in partial fulfillment of the
requirements for the degree of
Master of Engineering

Abstract

Techniques for developing low-complexity, robust Digital Signal Processing (DSP) algorithms with low-power consumption have become increasingly important. This thesis explores a framework, discrete-time randomized sampling, which allows the design of algorithms that meet some desired complexity, robustness or power constraints. Three distinct sampling schemes are presented based on randomized sampling of the input, of the filter impulse response, and iterative randomized sampling of the filter impulse response. Additive noise models are derived for each approach and error characteristics are analyzed for both white and colored sampling processes. It is shown that semi-white iterative randomized sampling leads to better error characteristics than white sampling through higher SNR as well as noise shaping. Discrete-time randomized sampling is then used as a filter approximation method and conditions are derived under which a randomized sampling approximation to the Wiener filter is guaranteed to lead a smaller mean-square estimation error than the best constrained LTI approximation. A low-power formulation for randomized sampling is also given and the tradeoff between power consumption and output quality is examined and quantified. Finally, this framework is used to model a class of random hardware failures and two algorithms are presented that guarantee a desired SNR level at the output under a given probability of hardware failure.

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Electrical Engineering

Acknowledgments

First and foremost, I would like to thank my thesis advisor, Al Oppenheim, for proving to me and convincing me that mayan just-in-time computation is inefficient since there is no processor idling time and therefore throughput is not constant. Al's mentorship and friendship have played a tremendous role in nurturing my interests in research and teaching and I am looking forward to many more years of stimulating interaction and personal growth. Al, thanks for everything, now we're ready to begin!

I am also grateful to Li Lee and Albert Chan for many hours of valuable technical discussions. My thanks go to Joe Saleh, Hur Koser, Lukasz Weber, Diego Syrowicz, and Jason Oppenheim for closely following the thesis progress.

The unconditional unlimited love of my family is a treasure that I will cherish forever. This thesis, like any other accomplishment, would have never been possible without their support. My deepest thanks also go to my great uncle, Wafic Said, for his constant support and belief in me.

Finally, I was extremely fortunate to get to know and work with Gary Kopec the summer of 1997 at Xerox PARC. The two months I interacted with Gary had a tremendous impact on me. His focus, determination, and joie de vivre will always be engraved in my mind and I will always regret not going back to PARC the following summer. Gary, you will always be in our hearts, this thesis is dedicated to you.

In memory of Gary E. Kopec

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Signal processing algorithms are designed to satisfy a variety of constraints depending on the application and the resources available. For example, when the amount of computation or the processing time is an issue, algorithms should be optimized for computational or time efficiency. Low-power algorithms, on the other hand, are used in environments where power is limited such as battery-powered mobile devices. Other situations present the need for robust algorithms in order to maintain a given performance level during hardware failure. Recently, with the rising demand for small mobile devices, there has been an increasing need for efficient, reliable, and low-power signal processing algorithms. In this thesis, we define a framework, discrete-time randomized sampling, with potential applications in the areas of hardware failure modeling, efficient algorithm development, and low-power signal processing. By randomly sampling signals or filters, this approach allows the modification of pre-existing algorithms to meet certain complexity, robustness or power constraints.

Randomized sampling is defined within the broader randomized signal processing framework used when the signal conversion or processing procedures are performed stochastically. Randomized signal processing is therefore fundamentally different from statistical signal processing where randomness is only used as a representation of signals. When sampling is performed at random time-intervals, it is described as randomized. Most of the randomized sampling literature starts from a continuous-time signal and thus consists of a randomization of the sampling process. Bilinskis and Mikelsons [2] survey the current results and issues in this area. By dealing with randomness exclusively in discrete-time, the framework presented by this thesis is distinct from the common approach found in the literature. Discrete-time randomized sampling could therefore be thought of as randomized down-sampling.

The approach developed in this thesis leads directly to three major applications in the areas of efficient convolution, low-power signal processing and hardware failures. In the remainder of this chapter, we present some relevant background in each of these areas. We then conclude with an outline of the thesis.

1.1 Efficient Convolution

Linear time-invariant (LTI) systems are widely used in signal processing applications, mainly because they can be analyzed in great detail and present good models of many physical phenomenon. A major advantage of discrete-time LTI systems is their ability to be fully characterized by the convolution sum. Implementing a filter is thus equivalent to implementing a convolution, and since processing power is usually an issue, efficient implementations tend to be favored. In general, convolution can be implemented either in the time or frequency domains. The fast Fourier transform (FFT) algorithm made the frequency-domain implementation of convolution much more computationally attractive since the implementation of a length N convolution using block processing and fast Fourier transforms requires an order of $NlogN$ multiplications per block of input, while the same operation implemented directly in time requires N^2 multiplications. However, frequency-domain processing leads to delays of the order of the filter length which is undesirable in some practical applications such as echo cancellation. In fact, many current Digital Signal Processing (DSP) chips implement convolution in time which can result in a very large computational complexity in applications where the filters have several thousands of coefficients. There have been several approaches to deal with this increased complexity. One class of approaches generates algorithms with exact solutions, i.e. no errors are introduced. Examples of such approaches include the parallel-decomposition algorithm of Wang and Siy [20] and the non-uniform partitioned block frequency-domain adaptive filter of Egelmeers and Sommen [7]. While these methods can be attractive because they do not introduce any distortions, they often either require specialized hardware or involve complex filter manipulations.

Simpler efficient algorithms can usually be obtained, at the cost of errors, from filter approximation techniques which reduce the filter length, and therefore reduce the amount of multiplications and additions in the convolution. Two common approximation techniques are the window method and the Parks-McClellan algorithm. While both tend to be used to obtain FIR filters from IIR filters, these procedures can also be used to approximate long FIR filters with shorter ones. By truncating the original filter, the rectangular window method provides the best mean-square approximation to a desired frequency response for a given filter length, i.e. it minimizes $\sum_n h_e^2[n]$ where $h_e[n]$ is the error filter which is defined as the difference between the original filter and the truncated filter. The Parks-McClellan algorithm, on the other hand, minimizes the maximum magnitude of the error filter, i.e. it minimizes $\mathrm{Max}(|H_e(e^{j\omega})|)$.

In this thesis, we present randomized sampling as an alternative filter approximation technique and, therefore, a method for efficient computation. In fact, by randomly setting some of the filter coefficients to zero, randomized sampling can be used as a procedure to perform only a subset of the multiplications in the convolution sum. An investigation of this technique will be carried out in Chapter 3.

1.2 Low-Power Signal Processing

Another area of interest in signal processing, power consumption, gained more attention in the early 90's due to the advances in storage and display technologies. These advances significantly reduced the power consumption due to the transfer of data and, as a result, increased the proportional power consumption by the processing modules [14]. A current major concern when developing signal processing algorithms for portable devices is to minimize the total power consumption in order to maximize the run-time while minimizing battery size [5]. In general, power can be minimized at four different levels: technology, circuit approaches, architectures and algorithms. Various techniques have been developed to reduce power consumption in digital circuits and research in this area is gaining tremendous attention. Brodersen *et al.* [3], present a good discussion of the relative gain to be expected from each of these areas while Evans and Liu [8] survey several low-power implementation strategies for common processing modules.

While both peak and time-averaged power are important factors in low-power systems design, battery size and weight are only affected by the average power. The time-averaged power consumption in CMOS digital circuits can be divided into four different components due to short-circuit, leakage, static, and switching as shown in equation 1.1 [13].

$$P_{\text{average}} = P_{\text{short-circuit}} + P_{\text{leakage}} + P_{\text{static}} + P_{\text{switching}} \tag{1.1}$$

All four components are affected by the supply voltage, V_{dd}. In addition, the first three components depend on the short-circuit, leakage, and static currents respectively while the last component depends on the load capacitor, C_L, the operating or switching frequency, f_s, and a constant, p, also known as the activity factor, which represents the probability of a power consuming transition. Table 1.1 gives mathematical expressions for all four components.

Short-circuit power	$P_{\text{short-circuit}} = I_{\text{short-circuit}} V_{dd}$
Static power	$P_{\text{static}} = I_{\text{static}} V_{dd}$
Leakage power	$P_{\text{leakage}} = I_{\text{leakage}} V_{dd}$
Switching power	$P_{\text{switching}} = p C_L V_{dd}^2 f_s$

Table 1.1: Individual power components contributing to the overall time-averaged power consumption.

In general, the switching power component is responsible for more than 90% of the total average power consumption [3]. As a result, switching power has gained great attention in low-power design. In general, algorithm design has little or no effect on power consumption resulting from leakage, short-circuit, or static, but it can play a major role in reducing the power due to switching. To first order, the average switching power consumption of a signal processing task can be written as:

$$\tilde{P}_{\text{switching}} = \sum_i N_i C_i V_{dd}^2 f_s \tag{1.2}$$

where C_i is the average capacitance switched per operation of type i corresponding to addition, multiplication, storage, or bus access. N_i is the number of operations of type i performed per output sample. V_{dd} is the operating supply voltage while f_s is the switching frequency. Traditionally, the switching frequency is fixed in many signal processing applications [6] and C_i is adjusted at the physical or circuit level. As a result, algorithm optimization for low-power processing has been mainly concerned with minimizing N_i. Several filter approximation methods have been used to explore low-power algorithms. Ludwig *et al.* [12], for example, use approximate processing to adaptively reduce the number of operations switched per sample based on signal statistics. They propose an adaptive filtering technique applied to frequency selective filters which dynamically chooses the lowest filter order that is guaranteed to achieve a preset output SNR. Pan [18], on the other hand, reduces power dissipation using approximate processing by taking advantage of the nature of audio signals. He uses adaptive filtering algorithms to reduce power by dynamically minimizing digital filter order. Chapter 4 of this thesis presents an algorithmic approach to low-power based on randomized sampling. Considerable power savings are obtained by reducing N_i, V_{dd}, and f_s in equation 1.2.

1.3 Hardware Failures

With the increase in dependability on digital systems, robustness is yet another issue to address when designing signal processing algorithms. Even with the use of high quality components and design techniques, there is always a non-negligible probability of system failure: robust algorithms should therefore be designed to deal with such failures. Most of the work in this area has been in the development of fault tolerant systems which are achieved through redundancy in hardware, software, information and/or computation [17]. Jiang *et al.* [10], for example, use adaptive fault tolerance for FIR filters to compensate for coefficient failures in an FIR adaptive filter by adding extra coefficients. Because of cost constraints, most of the fault tolerance strategies tend to address only single, statistically independent faults. However, multiple faults are very common especially with complex integrated systems and thus are receiving increased attention. Another disadvantage of the fault tolerance strategy is that it usually leads to very complex and inefficient algorithms. This can be particularly undesirable if only an acceptable, and not necessarily exact, performance is needed. An alternative approach to fault tolerance briefly explored by this thesis is the design of robust algorithms that are guaranteed to maintain a given signal-to-noise ratio in the presence of some types of hardware failures. This approach, while it does not guarantee a fault-free system, is generally much simpler and can be implemented efficiently. Chapter 5 lays out the hardware failure model and presents techniques for maintaining a desired level of performance under some types of hardware failures.

1.4 Thesis Outline

The thesis is organized as follows: in Chapter 2 we define the discrete-time randomized sampling framework by presenting three distinct sampling algorithms. Error models and properties are also derived for each sampling procedure. Discrete-time randomized sampling is then presented in an approximate filtering context in Chapter 3. Performance measures are defined and the randomized sampling algorithms are compared to LTI approximate processing techniques such as the rectangular window method. A low-power digital filtering formulation for discrete-time randomized sampling is given in Chapter 4. Power savings are evaluated and the trade-off between power and system performance is analyzed. Chapter 5 defines hardware failure models where the algorithms described in Chapter 2 are representative of the distortion introduced by the faulty system. As a result, techniques are derived to guarantee a desired performance level under a given probability of failure. Finally, Chapter 6 concludes by summarizing the contributions of the thesis and further research directions are suggested.

Chapter 2

Discrete-Time Randomized Sampling

In this chapter, we define discrete-time randomized sampling and present three different sampling approaches in the context of LTI filtering. We then derive linear noise models for each case and give second-order characteristics for the errors resulting from both white and colored sampling processes. Further error analysis is then presented by examining the in-band SNR obtained from each approach.

2.1 Definition

Discrete-time randomized sampling (DTRS) refers to any process which randomly sets some of the sample points of a discrete-time signal or filter impulse response to zero. DTRS is defined in the context of a general LTI system where the impulse response, $h[n]$, is deterministic and the input and output signals, $x[n]$ and $y[n]$, are wide-sense-stationary stochastic processes. The sampling can essentially be applied to the input, $x[n]$, or the filter, $h[n]$.

Sampling the input consists of randomly setting some of the input signal samples to zero. This can be viewed as a randomized down-sampling procedure where the randomization is at the signal conversion level. It is modeled as a multiplication by a non-zero-mean wide-sense-stationary random signal $r_s[n]$, independent of $x[n]$, that can only take on values 0 or

1. The resulting output, $y_{s1}[n]$, can then be written as:

$$y_{s1}[n] = (x[n]r_s[n]) * h[n] = \sum_{k=-\infty}^{+\infty} x[k]r_s[k]h[n-k] \tag{2.1}$$

In the remainder of this thesis, this procedure will be referred to as **DTRS of the input**.

DTRS can also be applied to the filter impulse response. Sampling the filter can be performed in two ways. First, a one-pass sampling scheme can be applied in which the filter is sampled only once and then the resulting filter is used to process the signal, $x[n]$. We refer to this sampling scheme as **DTRS of the impulse response**. It consists of randomly setting some of the filter coefficients to zero which makes the processing procedure stochastic. Again, the sampling process, $r_s[n]$, is a non-zero-mean wide-sense-stationary random signal and independent of $x[n]$. The output, $y_{s2}[n]$, can then be written as:

$$y_{s2}[n] = (h[n]r_s[n]) * x[n] = \sum_{k=-\infty}^{+\infty} h[k]r_s[k]x[n-k]. \tag{2.2}$$

A more elaborate sampling procedure for the impulse response consists of randomly sampling the filter at each iteration of the convolution sum. We refer to this sampling scheme as **iterative DTRS of the impulse response**, or simply **IDTRS**. The resulting filter is time-varying and stochastic, and can be modeled as multiplying the impulse response by a two-dimensional random signal in which one dimension corresponds to time and the second dimension is the ensemble member. The sampling process, $r_s[n;k]$, is a two-dimensional non-zero-mean wide-sense-stationary random signal and independent of $x[n]$. The output, $y_{s3}[n]$, can then be written as:

$$\begin{aligned} y_{s3}[n] &= x[n] * (r_s[n;k]h[n]) & (2.3) \\ &= \sum_{k=-\infty}^{+\infty} r_s[n;k]h[k]x[n-k]. & (2.4) \end{aligned}$$

2.2 Linear Noise Models

For each sampling method and for the case where the sampling is performed using wide-sense-stationary sampling sequences, one can derive linear noise models for the resulting distortion by rewriting the sampling process, $r_s[n]$, as the mean, m_r, added to a zero-mean process, $\hat{r}_s[n]$. Similarly the two-dimensional process, $r_s[n;k]$, can be rewritten as a two-dimensional zero-mean random process, $\hat{r}_s[n;k]$, added to the mean, m_r, where the two-dimensional mean is defined as $E(r_s[n;k]) = m_r$. As a result, each of the above sampling procedures can be rewritten as a scaled output added to noise. In order to gain some understanding as to the nature of the distortion introduced, second-order characterizations of the noise processes for each procedure are derived below.

2.2.1 DTRS of the Input Signal

Rewriting $r_s[n]$ as the sum of $\hat{r}_s[n]$ and m_r, equation (2.1) can be expressed as a scaled output added to a noise term:

$$\begin{aligned} y_{s1}[n] &= h[n] * ((\hat{r}_s[n] + m_r)x[n]) && (2.5) \\ &= m_r y[n] + e_1[n] && (2.6) \end{aligned}$$

where it can be shown that $m_r y[n]$ and $e_1[n]$ are uncorrelated (Appendix A). Furthermore, assuming $\hat{r}_s[n]$ is white, then it can be shown that:

$$R_{e_1e_1}[n] = m_r(1 - m_r)R_{xx}[0]R_{hh}[n] \qquad (2.7)$$

where $R_{e_1e_1}[n]$ and $R_{xx}[n]$ denote the autocorrelation functions of the error and the input signals respectively, and $R_{hh}[n]$ is the deterministic autocorrelation of the impulse response. This method can thus be modeled as adding white noise to the scaled input signal and then filtering the resulting signal with $h[n]$ as shown in Figure 2-1. $w_1[n]$ is a zero-mean wide-sense-stationary white noise process with variance $m_r(1 - m_r)R_{xx}[0]$.

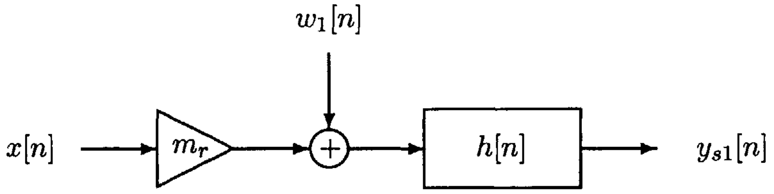

Figure 2-1: Linear Model for Discrete-Time Randomized Sampling of the Input.

2.2.2 DTRS of the Impulse Response

Following the same approach as in the previous section, $y_{s2}[n]$ can be expressed as a scaled output added to a noise term:

$$\begin{aligned} y_{s2}[n] &= x[n] * ((\hat{r}_s[n] + m_r)h[n]) && (2.8) \\ &= m_r y[n] + e_2[n]. && (2.9) \end{aligned}$$

Again, it can be shown that $m_r y[n]$ and $e_2[n]$ are uncorrelated. In addition, assuming $\hat{r}_s[n]$ is white, then the autocorrelation of the error, $R_{e_2e_2}[n]$, is given by:

$$R_{e_2e_2}[n] = m_r(1 - m_r)R_{xx}[n]R_{hh}[0]. \qquad (2.10)$$

This method can thus be represented as adding colored noise to the scaled output, as shown in Figure 2-2. The noise spectrum has the same shape as the input power spectrum: $w_2[n]$ is a zero-mean colored noise with autocorrelation $m_r(1 - m_r)R_{hh}[0]R_{xx}[n]$.

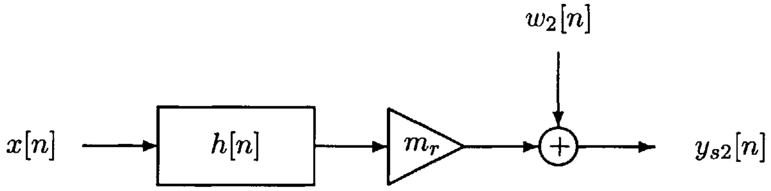

Figure 2-2: Linear Model for Discrete-Time Randomized Sampling of the Impulse Response.

2.2.3 Iterative DTRS of the Impulse Response

We can rewrite $y_{s3}[n]$ as :

$$\begin{aligned} y_{s3}[n] &= x[n] * ((\hat{r}_s[n;k] + m_r)h[n]) && (2.11) \\ &= m_r y[n] + e_3[n]. && (2.12) \end{aligned}$$

Again $m_r y[n]$ and $e_3[n]$ are uncorrelated. In addition, we assume that $\hat{r}_s[n;k]$ is two-dimensional white, so that the autocorrelation $R_{\hat{r}_s\hat{r}_s}[n;k]$ of $\hat{r}_s[n;k]$ is given by:

$$R_{\hat{r}_s\hat{r}_s}[n;k] = m_r(1 - m_r)\delta[n;k] = m_r(1 - m_r)\delta[n]\delta[k] \qquad (2.13)$$

and that $x[n]$ is a wide-sense-stationary random process. Then the autocorrelation of $e_3[n]$, $R_{e_3e_3}[n]$, is given by:

$$R_{e_3e_3}[n] = m_r(1 - m_r)R_{xx}[0]R_{hh}[0]\delta[n]. \tag{2.14}$$

This method can thus be represented as adding white noise to the scaled original output resulting from filtering the input signal, $x[n]$, with $h[n]$ as shown in Figure 2-3. $w_3[n]$ is zero-mean wide-sense-stationary white noise with variance $m_r(1 - m_r)R_{hh}[0]R_{xx}[0]$.

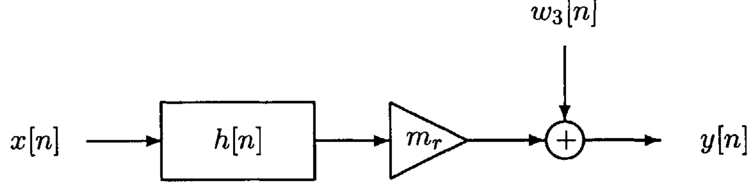

Figure 2-3: Linear Model for Iterative Discrete-Time Randomized Sampling of the Impulse Response.

2.3 Error Analysis

The previous section showed that if the sampling process is wide-sense-stationary, the output from each of the three sampling procedures could be rewritten as the original output scaled by the mean of the sampling process added to an error term which was shown to be wide-sense-stationary and uncorrelated with the signal. Table 2.1 summarizes the second-order characteristics of the errors obtained when the sampling process is white. All three errors have the same total power or variance since

$$R_{e_1e_1}[0] = R_{e_2e_2}[0] = R_{e_3e_3}[0] = m_r(1 - m_r)R_{xx}[0]R_{hh}[0]. \tag{2.15}$$

However, they each have different spectral densities.

The signal-to-noise ratio (SNR), defined as the ratio of signal variance (power) to noise variance, is commonly used as a measure for the amount of degradation of a signal by additive noise [1]. A mathematical expression for the SNR is given in equation (2.16). It is clear that, since all errors have the same total power, the three sampling methods are equivalent if the only criteria is to maximize SNR. However, using the SNR as defined in equation (2.16) is sometimes misleading since a more relevant measure is typically the noise

Method	Error	Autocorrelation Function
DTRS of $x[n]$	$e_1[n]$	$R_{e_1e_1}[n] = m_r(1-m_r)R_{xx}[0]R_{hh}[n]$
DTRS of $h[n]$	$e_2[n]$	$R_{e_2e_2}[n] = m_r(1-m_r)R_{hh}[0]R_{xx}[n]$
Iterative DTRS of $h[n]$	$e_3[n]$	$R_{e_3e_3}[n] = m_r(1-m_r)R_{xx}[0]R_{hh}[0]\delta[n]$

Table 2.1: Autocorrelation Functions of the Errors obtained from each Sampling Scheme using a White Sampling Process.

power in relation to the signal spectrum. Therefore, a more useful measure for the amount of distortion is the in-band SNR defined as the ratio of the power in the output signal to the noise power within the signal band. Equation (2.17) gives a mathematical expression for this measure and Table 2.2 lists the in-band signal-to-noise ratios for the cases when $|H(e^{j\omega})|^2$ is bandlimited and when $\Phi_{xx}(e^{j\omega})$ is bandlimited.

$$\text{SNR} = \frac{\text{variance of signal}}{\text{variance of noise}} = \frac{m_r^2 R_{yy}[0]}{R_{ee}[0]} \tag{2.16}$$

$$\text{SNR}_{in} = \frac{\text{signal power}}{\text{noise power in signal band}} = \frac{m_r^2 R_{yy}[0]}{\frac{1}{2\pi}\int_{<\text{signal band}>} \Phi_{ee}(e^{j\omega})d\omega} \tag{2.17}$$

Method	Case 1: $\vert H(e^{j\omega})\vert^2$ is bandlimited to ω_0	Case 2: $\Phi_{xx}(e^{j\omega})$ is bandlimited to ω_0
DTRS of $x[n]$	$\text{SNR}_{in}^1 = \frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}$	$\text{SNR}_{in}^1 = \frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]\frac{1}{\pi}\int_0^{\omega_0}\vert H(e^{j\omega})\vert^2 d\omega}$
DTRS of $h[n]$	$\text{SNR}_{in}^2 = \frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{hh}[0]\frac{1}{\pi}\int_0^{\omega_0}\Phi_{xx}(e^{j\omega})d\omega}$	$\text{SNR}_{in}^2 = \frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}$
Iterative DTRS of $h[n]$	$\text{SNR}_{in}^3 = \frac{\pi}{\omega_0}\frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}$	$\text{SNR}_{in}^3 = \frac{\pi}{\omega_0}\frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}$

Table 2.2: In-band Signal-to-Noise Ratios for the Three Sampling Methods.

It is clear from Table 2.2 that, since $\frac{\pi}{\omega_0} > 1$, IDTRS always leads to a higher in-band signal-to-noise ratio than DTRS of the input if the filter is bandlimited. If, on the other hand, the input has a bandlimited power spectrum, then IDTRS always leads to a higher in-band signal-to-noise ratio than DTRS of the impulse response. In addition, it follows in a relatively straight-forward way that in the case where the filter is bandlimited to ω_0, if the ratio of input power within the filter band to the total power in the input is greater than $\frac{\omega_0}{\pi}$, then IDTRS always leads to a higher in-band SNR than DTRS of the filter, and thus in this case, IDTRS leads to the highest in-band SNR. In the case where the input power is bandlimited to ω_0, then it can be shown that IDTRS leads to a higher in-band signal-to-noise ratio than DTRS of the input if and only if the ratio of power in the filter within the input band to the total power in the filter is higher than $\frac{\omega_0}{\pi}$, and in this case, IDTRS leads to the highest in-band SNR. Table 2.3 summarizes these results.

Case 1 $\lvert H(e^{j\omega})\rvert^2 = 0$ for $\lvert\omega\rvert \geq \omega_0$	Case 2 $\Phi_{xx}(e^{j\omega}) = 0$ for $\lvert\omega\rvert \geq \omega_0$
$\mathrm{SNR}^1_{in} \leq \mathrm{SNR}^3_{in}$ and $\mathrm{SNR}^1_{in} \leq \mathrm{SNR}^2_{in}$	$\mathrm{SNR}^2_{in} \leq \mathrm{SNR}^3_{in}$ and $\mathrm{SNR}^2_{in} \leq \mathrm{SNR}^1_{in}$
In addition: $\mathrm{SNR}^2_{in} \leq \mathrm{SNR}^3_{in}$ if and only if: $\frac{\int_0^{\omega_0} \Phi_{xx}(e^{j\omega})d\omega}{\int_0^{\pi} \Phi_{xx}(e^{j\omega})d\omega} \geq \frac{\omega_0}{\pi}$	In addition: $\mathrm{SNR}^1_{in} \leq \mathrm{SNR}^3_{in}$ if and only if: $\frac{\int_0^{\omega_0} \lvert H(e^{j\omega})\rvert^2 d\omega}{\int_0^{\pi} \lvert H(e^{j\omega})\rvert^2 d\omega} \geq \frac{\omega_0}{\pi}$

Table 2.3: In-band SNR Comparisons for the Three Sampling Methods.

2.4 Non-White Sampling

While in previous sections, we assumed the sampling process was white, in this section, we relax this condition and explore non-white sampling in the context of iterative randomized sampling of the impulse response. In particular, having a correlated sampling process could potentially lead to better error characteristics through lower error variance or higher in-band signal-to-noise ratio. In the following, we examine cases where the two-dimensional sampling process, $\hat{r}_s[n;k]$, is white in one dimension and colored in the other. Specifically, we first investigate the case where $\hat{r}_s[n;k]$ is white in the n-dimension and arbitrary in the k-dimension, we then proceed to the case where $\hat{r}_s[n;k]$ is colored in the n-dimension and white in the k-dimension.

2.4.1 IDTRS using a Correlated Sampling Process at each Convolution Step

Consider the case where the processes used to sample the impulse response in order to compute successive output samples are uncorrelated with each other. However, the process used to compute each output sample has some correlation, i.e. is not white. Then, assuming the mean of the process is m_r, the autocorrelation function of the sampling process can be written as:

$$R_{\hat{r}_s\hat{r}_s}[m;k] = (1-m_r)m_r\delta[m]R_{gg}[k] \tag{2.18}$$

where $R_{gg}[k]$ is a one-dimensional arbitrary autocorrelation function which describes the second-order statistics of $\hat{r}_s[n;k]$ in the k-dimension. We can then show (Appendix B) that the autocorrelation of the error, $e_3[n]$, can be written as:

$$R_{e_3e_3}[m] = m_r(1-m_r)\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{gg}[l-k]R_{xx}[l-k]\delta[m] \tag{2.19}$$

i.e. the error resulting from such sampling is still white however the variance is now different from the sampling using a two-dimensional white process. In order to gain some intuition, consider the special case where $R_{gg}[m] = \delta[m] - \alpha\delta[m-1] - \alpha\delta[m+1]$ where $-1 < \alpha < 1$, i.e. where there is correlation at only one lag. In this case, equation (2.19) becomes:

$$R_{e_3e_3}[m] = m_r(1-m_r)(R_{hh}[0]R_{xx}[0] - 2\alpha R_{hh}[1]R_{xx}[1])\delta[m]. \tag{2.20}$$

The variance is no longer only a function of the energy in the filter and the input, as was the case for white sampling, it now also depends on the energy at one lag in the filter and the input. In particular, if $R_{hh}[1]$ and $R_{xx}[1]$ have the same sign and α is positive, or if $R_{hh}[1]$ and $R_{xx}[1]$ have opposite signs and α is negative, the resulting error has a lower variance than the one resulting from randomized sampling using a two-dimensional white sequence. This therefore suggests a method for adjusting the correlation of the sampling process in order to reduce the error variance if some information about the signal and filter is known.

2.4.2 IDTRS using a Sampling Process Correlated across Output Samples

Now consider the case where at each convolution step, the sampling process is white, however the processes used in computing different output time samples have some non-zero correlation. Then, if the process mean is m_r, the two-dimensional autocorrelation function of the sampling process can be written as:

$$R_{\hat{r}_s\hat{r}_s}[m;k] = m_r(1-m_r)R_{vv}[m]\delta[k] \tag{2.21}$$

where $R_{vv}[m]$ is a one-dimensional arbitrary autocorrelation function which describes the second-order statistics of $\hat{r}_s[n;k]$ in the n-dimension. We can then show (Appendix B) that the autocorrelation of the error $e_3[n]$, can be written as:

$$R_{e_3e_3}[m] = m_r(1-m_r)R_{hh}[0]R_{vv}[m]R_{xx}[m]. \tag{2.22}$$

The error, in this case, is no longer white and the correlation is determined by the amount of correlation in the sampling process as well as in the input. Note that in the limiting case where $R_{vv}[m]$ is 1 for all m, $R_{e_3e_3}[m] = R_{e_2e_2}[m]$, i.e. the autocorrelation function equals the autocorrelation function resulting from DTRS of the impulse response. This should not be surprising since in the limiting case, both sampling procedures are equivalent up to second-order statistics, and as a result, the errors should have the same autocorrelation functions. Furthermore, consider the case where $R_{vv}[m] = \delta[m] - \alpha\delta[m-1] - \alpha\delta[m+1]$ where again $-1 < \alpha < 1$, then equation (2.22) can be rewritten as:

$$R_{e_3e_3}[m] = m_r(1-m_r)R_{hh}[0]R_{xx}[0](\delta[m] - \alpha\frac{R_{xx}[1]}{R_{xx}[0]}\delta[m-1] - \alpha\frac{R_{xx}[1]}{R_{xx}[0]}\delta[m+1]). \tag{2.23}$$

If $R_{xx}[1]$ and α have the same sign, then the error power is highest at high frequencies whereas if $R_{xx}[1]$ and α have opposite signs, then the error power is highest at low frequencies. In fact, equation (2.23) shows that the error exhibit spectral shaping reminiscent of the error behavior in a sigma-delta Analog-to-Digital converter. As a result, this method can be used to shape the error and increase in-band SNR.

2.5 Summary

This chapter defined discrete-time randomized sampling as a method of setting some of the samples of a discrete-time signal or filter to zero. Three methods for performing the randomized sampling were examined and linear noise models were derived for the case when the sampling process was white. Further error analysis in the case of white sampling was then presented and conditions were derived where iterative DTRS of the impulse response always outperformed the other sampling methods. Finally, two cases of non-white sampling were investigated and the resulting errors were analyzed. The observed lower error variance and noise shaping suggested trading off structure in the sampling process for improved error characteristics.

Chapter 3

Filter Approximation Based on Randomized Sampling

Approximate Signal Processing was first defined by Nawab *et al.* [16] as a formal approach of trading-off quality for resources. In this Chapter, we consider the use of randomized sampling for filter approximation and therefore for trading-off quality for computation.

In various signal processing applications, one is interested in reducing the total amount of computation required to perform a filtering operation. Such applications include reducing the amount of computation in order to speed up the processing time or reduce power consumption. Approximation techniques have been used as ways to reduce computation since these techniques reduce the filter length, and therefore reduce the amount of multiplications and additions in the convolution. A common LTI approximation technique consists of reducing the filter length using one of the optimal FIR filter design methods such as the window method or the Parks-McClellan algorithm. All LTI approximation techniques introduce an error that can be modeled as in Figure 3-1. In this figure, $h[n]$ represents the original desired filter of length P and $h_e[n]$ the error filter resulting from approximating $h[n]$ with a filter of length M where $M < P$. Discrete-time randomized sampling of the impulse response, as defined in the previous chapter, can be used as a filter approximation technique and a method for efficient computation since by randomly setting some of the filter coefficients to zero, randomized sampling can be used as a procedure to perform only a subset of the multiplications in the convolution sum if one checks for multiply-by-zero operations. This chapter investigates the potential of DTRS as a filter approximation tech-

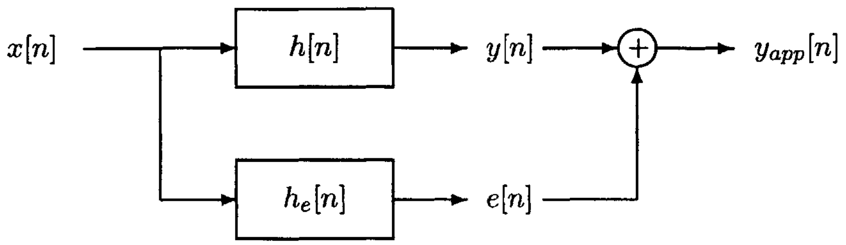

Figure 3-1: Block Diagram of the LTI Approximation of $h[n]$.

nique by comparing the error introduced through DTRS with errors introduced through LTI filter approximation techniques such as the rectangular window method. Error measures and specific classes of inputs are defined and explored.

3.1 Approximate Filtering Formulation

In Chapter 2, we noted that both non-iterative and iterative randomized sampling of the impulse response lead to errors with equal variances. As a result, if the performance measure is chosen to be the output error variance, we will only analyze the IDTRS case since the non-iterative case is equivalent. In addition, in Section 2.3, we showed that if the filter is bandlimited to ω_0 and the ratio of the input signal energy within the pass-band to the total input signal energy is greater than or equal to $\frac{\omega_0}{\pi}$, iterative sampling always outperforms non-iterative sampling if the in-band SNR is the performance measure. As a result, in the following, we will assume that these conditions are met and will only focus on iterative sampling of the impulse response. In addition, we will assume that the sampling is performed using a two-dimensional wide-sense-stationary white process.

Since IDTRS in this case results in the original output scaled down by the mean of the sampling process added to white noise, we first need to re-scale the output by $1/m_r$ in order to formulate the problem in the context of Figure 3-1. The rescaling is shown in Figure 3-2, and results in an additive noise model where the error variance is now scaled by $1/m_r^2$, i.e. we can rewrite $y_s[n]$ as:

$$y_s[n] \quad = \quad y[n] + e[n] \tag{3.1}$$

where the output and the noise are uncorrelated and the autocorrelation of $e[n]$, $R_{ee}[n]$, is

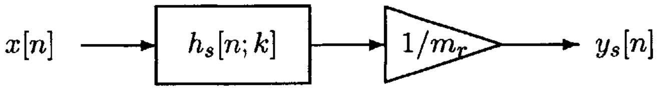

Figure 3-2: Re-Scaled IDTRS. $h_s[n;k]$ is the sampled version of $h[n]$ as explained in Chapter 2 and m_r is the mean of the sampling process.

given by:

$$R_{ee}[n] = \frac{(1-m_r)}{m_r} R_{xx}[0] R_{hh}[0] \delta[n]. \tag{3.2}$$

Figure 3-2 and equations (3.1) and (3.2) summarize the approximate filtering formulation of discrete-time randomized sampling. These results can now be used to compare randomized sampling to other approximation techniques in the context of LTI filtering. An example of approximate filtering is given in the next section where the randomized sampling approach is compared to other approximation approaches.

3.2 Linear Minimum Mean-Square Estimation with Complexity Constraints

In many cases, a filter is obtained by minimizing the mean-square difference between the output of the filter, $\hat{y}[n]$, and some desired signal, $y[n]$. Such applications include detection, prediction, and estimation. In this section we consider the problem of approximating the linear minimum mean-square estimator in order to reduce complexity where complexity is given by the amount of computation needed to obtain the estimate. The problem can be formulated as follows: we wish to obtain an approximation to the best linear mean-square estimator of the signal $y[n]$ based on some received data, $r[n]$. The best linear estimator, in the mean-square sense, is the Wiener filter. If $\hat{y}[n]$ denotes the output of the Wiener filter, then, by definition, the mean-square estimation error, $E((\hat{y}[n]-y[n])^2)$, is minimized, and as a result, the estimation error, $(\hat{y}[n]-y[n])$, is orthogonal to any linear function of the data, i.e. $E(A\{r[n]\}(\hat{y}[n]-y[n]))=0$, for *any* linear operator $A\{\}$. In the following, we consider two different approximation techniques to the Wiener filter. The first uses IDTRS as the approximation method while the second uses an LTI approximation technique with distortion modeled by Figure 3-1. We wish to define conditions under which the IDTRS approximation leads to a smaller increase in estimation error than the LTI approximation. Specifically, if ϵ_{rs}^2 designates the mean-square estimation error resulting from the IDTRS approximation method, and ϵ_w^2 is the mean-square estimation error resulting from the LTI approximation method, we would like to derive conditions such as equation (3.3) below holds.

$$\epsilon_{rs}^2 < \epsilon_w^2 \tag{3.3}$$

In the following, we will refer to the difference between the output of the original Wiener filter and the output of the approximated filter as the *approximation error* whereas the *estimation error* refers to the difference between the desired signal and the output of the estimator.

3.2.1 Mean-Square Estimation Error

In this section we derive expressions for the mean-square estimation error for each approximation technique. In general, all approximation techniques can be represented through an additive error model where the output to the approximated Wiener filter is $\tilde{y}[n] + e[n]$ where $e[n]$ is white and uncorrelated with $\tilde{y}[n]$ for the IDTRS method, while $e[n]$ is colored and correlated with $\tilde{y}[n]$ for the LTI method. The mean-square estimation error is therefore given by:

$$\begin{aligned}
\epsilon^2 &= E((\tilde{y}[n] + e[n] - y[n])^2) && (3.4)\\
&= E((\tilde{y}[n] - y[n])^2) + E(e^2[n]) + E(e[n](\tilde{y}[n] - y[n])) && (3.5)\\
&= \epsilon_{\text{wiener}}^2 + \sigma_e^2 + E(e[n](\tilde{y}[n] - y[n])) && (3.6)\\
& && (3.7)
\end{aligned}$$

where $\epsilon_{\text{wiener}}^2$ is the minimum mean-square estimation error resulting from the Wiener filter and σ_e^2 is the approximation error variance (note all signals are zero-mean). As a result, the increase in the mean-square estimation error, Δ_{ϵ^2} is given by:

$$\Delta_{\epsilon^2} = \sigma_e^2 + E(e[n](\tilde{y}[n] - y[n])). \tag{3.8}$$

In Chapter 2, we showed that the approximation error resulting from randomized sampling is uncorrelated with the output signal, i.e.:

$$E(e_{rs}[n]\tilde{y}[n]) = 0. \tag{3.9}$$

It also directly follows that, since the randomizing process is independent of all other signals, $e_{rs}[n]$ is also uncorrelated with $y[n]$:

$$E(e_{rs}[n]y[n]) = 0. \tag{3.10}$$

The increase in mean-square estimation error for IDTRS is therefore simply the approximation error variance:

$$\Delta_{\epsilon^2_{rs}} = \sigma^2_{rs}. \tag{3.11}$$

In the LTI approximation case, the approximation error can be expressed as the convolution of the input to the filter (the data) with an error filter as shown in Figure 3-1. As a result, the approximation error is a linear combination of the input data and thus, using the orthogonality property of the minimum mean-square estimation error, we conclude:

$$E(e_w[n](\tilde{y}[n] - y[n])) = 0. \tag{3.12}$$

The increase in mean-square estimation error resulting from LTI approximation techniques is therefore equal to the variance of the LTI approximation error:

$$\Delta_{\epsilon^2_w} = \sigma^2_w. \tag{3.13}$$

Equation (3.3) is therefore equivalent to equation (3.14) below:

$$\sigma^2_{rs} = E(e^2_{rs}[n]) < \sigma^2_w = E(e^2_w[n]). \tag{3.14}$$

Expressions for σ^2_w and σ^2_{rs} are given in equations (3.15) and (3.16) respectively where $R_{rr}[m]$ is the autocorrelation function of $r[n]$ and $S_{rr}(e^{j\omega})$ its Fourier transform. $h_e[n]$ is the error filter as defined by Figure 3-1 and $h[n]$ is the Wiener filter.

$$\sigma_w^2 = \sum_k R_{rr}[k]R_{h_e h_e}[-k] = \frac{1}{2\pi}\int_{-\pi}^{\pi} S_{rr}(e^{j\omega})|H_e(e^{j\omega})|^2 \tag{3.15}$$

$$\sigma_{rs}^2 = \frac{1-m_r}{m_r}R_{rr}[0]R_{hh}[0] = \frac{1}{2\pi}\frac{1-m_r}{m_r}R_{rr}[0]\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 \tag{3.16}$$

3.2.2 Zero-Mean, Unit-Variance, White Process, $r[n]$

We consider the case where $r[n]$ is a zero-mean, unit-variance white process. In this case, σ_{rs}^2 and σ_w^2 can be rewritten as follows:

$$\sigma_w^2 = \frac{1}{2\pi}\int_{-\pi}^{\pi} |H_e(e^{j\omega})|^2 = \sum_k h_e^2[k] \tag{3.17}$$

$$\sigma_{rs}^2 = \frac{1}{2\pi}\frac{1-m_r}{m_r}\int_{-\pi}^{\pi} |H(e^{j\omega})|^2 = \frac{1-m_r}{m_r}\sum_k h^2[k] \tag{3.18}$$

Among all possible LTI approximation techniques which are modeled as Figure 3-1, the rectangular window method is the one that leads to the best mean-square approximation to the original frequency response [1], i.e. it minimizes the expression: $\frac{1}{2\pi}\int_{-\pi}^{\pi}|H_e(e^{j\omega})|^2 d\omega$ and therefore minimizes the value of σ_w^2 in equation (3.17). We will therefore use the rectangular window method as the LTI approximation technique and explore conditions under which randomized sampling leads to a better estimate.

In this case, equation (3.14) is satisfied if and only if:

$$0 < \frac{1-m_r}{2m_r-1} < \frac{\sum_k h_e^2[k]}{\sum_k h_w^2[k]} \tag{3.19}$$

where $h_w[n]$ is the windowed impulse response. The length M of the window is determined by m_r:

$$m_r = \frac{M}{P} \tag{3.20}$$

where P is the length of the original impulse response. The condition given by equation (3.19) states that if the ratio of the total energy in the portion of the impulse response not covered by the window to the total energy in the windowed impulse response is greater than $\frac{1-m_r}{2m_r-1}$, then iterative randomized sampling leads to an estimation error with lower variance than filter approximation using a rectangular window. In addition, the first inequality in equation (3.19) shows that IDTRS can never outperform LTI approximation techniques if $m_r < 0.5$. Figure 3-3 shows how the ratio $\frac{1-m_r}{2m_r-1}$ varies with m_r.

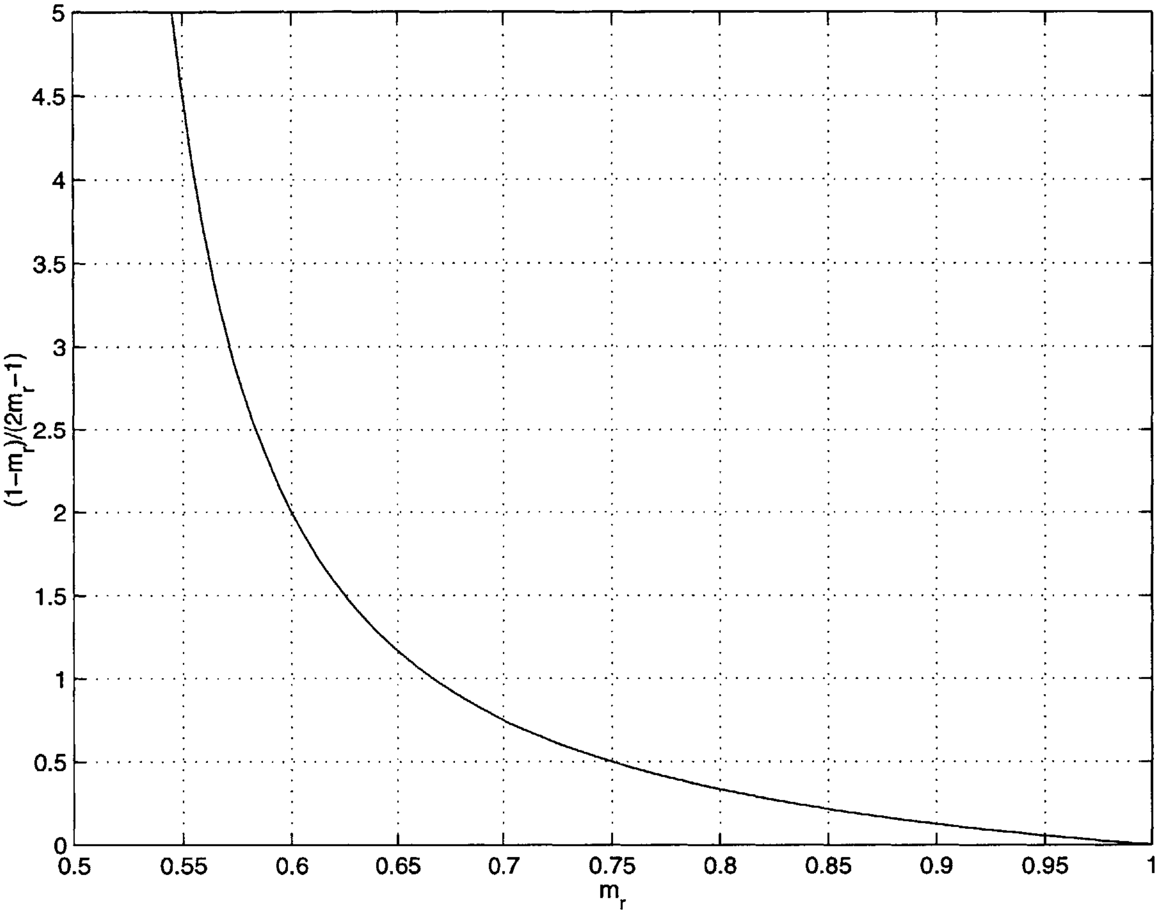

Figure 3-3: Boundary Condition for the LMMSE Approximation Problem. IDTRS Outperforms *any* LTI Approximation Technique if $\frac{\sum_k h_e^2[k]}{\sum_k h_w^2[k]} > \frac{1-m_r}{2m_r-1}$.

It is clear from Figure 3-3 that, since the length of the window is constrained by equation (3.20), and assuming the window is positioned optimally, i.e. around the region of the filter with maximal energy, equation (3.19) cannot be satisfied if $|h[n]|$ is monotonic or has a single peak. A direct consequence of this is that if the filter of interest is a linear-phase low-pass filter and the goal is to minimize error variance, then the window method is a better approximation method than randomized sampling. However, if the Wiener filter impulse response has multiple peaks, i.e. its energy is spread out in time, and the desired number of non-zero taps in the approximate filter, M, is such that equation (3.19) is satisfied, then it is better, in the mean-square sense, to use a long filter and iteratively randomly sample it than use the best mean-square linear filter of length M.

3.2.3 Non-White $r[n]$

The truncated Wiener filter is the best length-M linear minimum mean-square estimator only if the received signal, $r[n]$, is white. If $r[n]$ is not white, then the optimal linear mean-square estimator needs to be computed every time M changes. This may not be desirable or even feasible in cases where resources are limited or if algorithm simplicity is of primary concern. In these cases, a straight-forward approach would be to truncate the Wiener filter or iteratively randomly sample it if the conditions in equation (3.19) are satisfied. Alternatively, the estimator could be decomposed into two systems with different resource allocations: a whitening filter implemented accurately, and a constrained Wiener filter (with limited resources) operating on the white data. In this case, the problem is similar to the one discussed in the previous section and the IDTRS method may lead to a better mean-square estimate than the best LTI approximation method.

3.3 Summary

This chapter exploited randomized sampling as a filter approximation technique. Results derived in Chapter 2 were used to compare IDTRS filter approximations to LTI approximation techniques. Specifically, we compared the performance of both methods in the context of linear minimum mean-square estimation with complexity constraints was compared and conditions were derived, in the case of white data, where IDTRS lead to a better mean-square estimate than the best constrained linear mean-square estimator. Scenarios where it would be desirable to use IDTRS even if the data is not white were also discussed.

Chapter 4

Low-Power Digital Filtering

In this chapter, we explore the use of randomized sampling as a way to reduce power consumption in a filtering context by examining the tradeoff between power savings and output quality. To this end, we first give a low-power formulation for randomized sampling by showing how setting some filter coefficients to zero can lead to powering down parts of the processor and therefore reducing the switching activity. Further savings can be obtained by slowing-down the computation in order to avoid processor idling time, and therefore reducing the voltage supply. Clearly, the additional savings due to scaling the voltage supply are not specific to randomized sampling but a consequence of reduced computation and have been already thoroughly examined [6]. However, some of these results will be re-derived in this chapter in order to quantify the tradeoff between power savings and output quality for the case of randomized sampling.

4.1 Low-Power Formulation

With the recent expanding interest in wireless systems, reducing power consumption is becoming an additional design factor along with size and speed. As discussed in Chapter 1, algorithmic approaches to low-power design are mainly concerned with reducing the switching power given by equation (4.1)

$$\tilde{P}_{\text{switching}} = \sum_i N_i C_i V_{dd}^2 f_s \tag{4.1}$$

where C_i is the average capacitance switched per operation of type i corresponding to addition, multiplication, storage, or bus access. N_i is the number of operations of type i performed per output sample, V_{dd} is the operating supply voltage, and f_s is the switching frequency. In general, any discrete-time randomized sampling approach in a filtering context consists of reducing the average number of multiplications and additions performed per output sample. As a result, any discrete-time randomized sampling approach can be used in a low-power processing context where the power savings result from a reduction in N_i in equation (4.1).

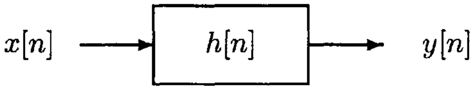

Figure 4-1: Original System.

Specifically, consider a time-domain implementation of the convolution illustrated by the system shown in Figure 4-1. If the filter, $h[n]$, has length p_0, then a time-domain implementation of the convolution requires $(p_0 + 1)$ multiplications and p_0 additions per output sample as can be seen in equation (4.2).

$$y[n] = h[0]x[n] + h[1]x[n-1] + h[2]x[n-2] + ... + h[p_0]x[n-p_0]. \tag{4.2}$$

Discrete-time randomized sampling of the impulse response can lead to power savings through both a lower switching activity (lower N_i) and a scaling of the voltage supply (lower V_{dd}) which we examine separately in the next sections.

4.1.1 Switching Activity Reduction

Both DTRS of the impulse response and IDTRS using a sampling process with mean m_r consist of setting an average of $(1 - m_r)$ filter coefficients to zero prior to computing each output sample. In the DTRS case, the same subset of filter coefficients is set to zero for all time whereas in the IDTRS case, the subset is different for each output sample. As a result, for both cases, $m_r(p_0 + 1)$ multiplications and $m_r p_0$ additions are required, on average, per output sample, i.e. the number of multiplications and additions used per output sample is reduced by a factor of $(1 - m_r)$ which corresponds to a similar reduction in the switching activity. Since average switching power is a linear function of the amount of computation required per output sample, a reduction by $(1 - m_r)$ in computation leads to a reduction in power consumption by the same factor.

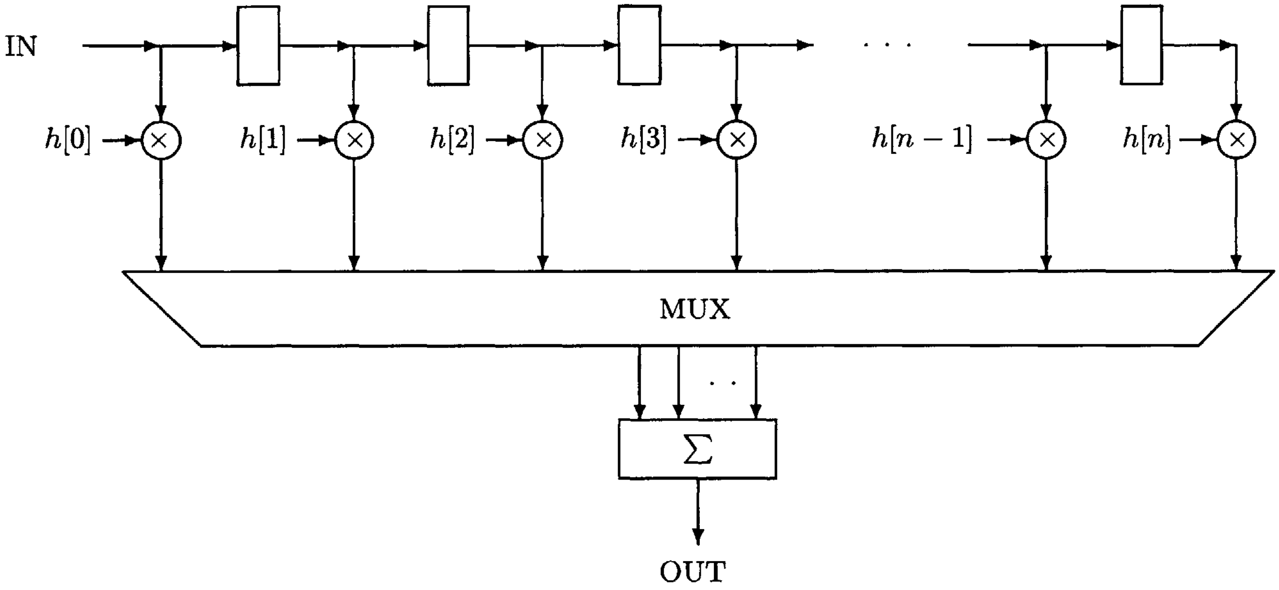

Figure 4-2: FIR Filter Structure for Low-Power Filtering Using Discrete-Time Randomized Sampling. MUX is used as an abstraction to a collection of multiplexers.

Figure 4-2 illustrates a low-power implementation for discrete-time randomized sampling. At each time point or clock cycle, a subset of the outputs of the multipliers is picked by a collection of multiplexers (abstracted by MUX in the figure), the remaining multiplications are not performed therefore reducing computation. The chosen subset is then used as an input to an accumulator which performs the final addition to generate the output. Figure 4-3 shows an equivalent implementation where the powering down of a subset of the multiplication branches is made explicit by the addition of switches for each branch. Branches that are powered down are not used by the adders. The normalized power consumption as a function of the mean, m_r, of the sampling process is shown in Figure 4-4.

4.1.2 Supply Voltage Reduction

Figure 4-3 suggests yet a further reduction in power through an increase in the propagation delay which allows a reduction in the supply voltage while maintaining the same throughput

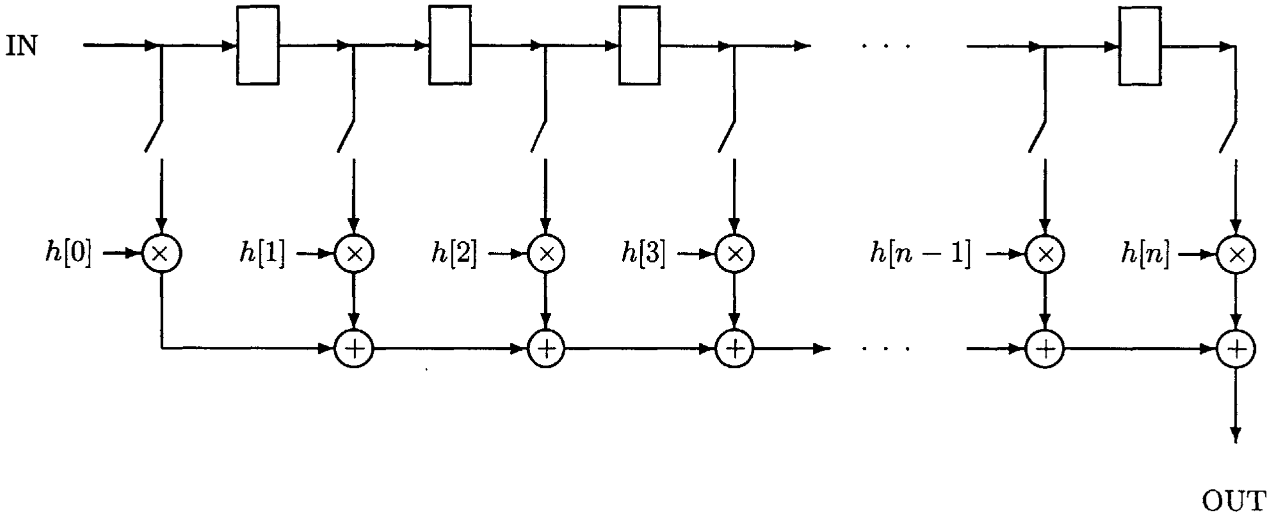

Figure 4-3: FIR Filter Structure for Low-Power Filtering using Randomized Sampling.

as the initial system. Burd and Brodersen [4] note that if the clock-rate is equal to the increase of the critical path delay, then throughput is proportional to the number of parallel operations per clock cycle (concurrency) divided by the delay, i.e.:

$$T = \frac{Operations}{second} = \gamma\frac{Concurrency}{Delay} \tag{4.3}$$

In this case, delay refers to the time it takes to perform a multiplication. Most digital signal processing applications are real-time applications and as a result the throughput is fixed. In the filtering context of Figure 4-1, the number of parallel operations per clock cycle correspond to the number of multiplications per output sample which is equal to $(p_0 + 1)$ in the original system. Using the randomized sampling approximation method, the number of multiplication is reduced to $m_r(p_0 + 1)$ and so is the amount of concurrency per clock cycle. It is thus clear from equation (4.3) that an increase in the delay by a factor of $1/m_r$ will compensate for the decrease in the amount of concurrency and therefore the same throughput could be maintained. In this context, there is no need to increase throughput, power reduction has higher priority. Chandrakasan *et al.* [6] have already explored the idea of lowering the voltage supply if less work needs to be done by developing an approach to minimize the energy dissipation per output sample in variable-load DSP systems. Here, we derive similar results and apply them to randomized sampling.

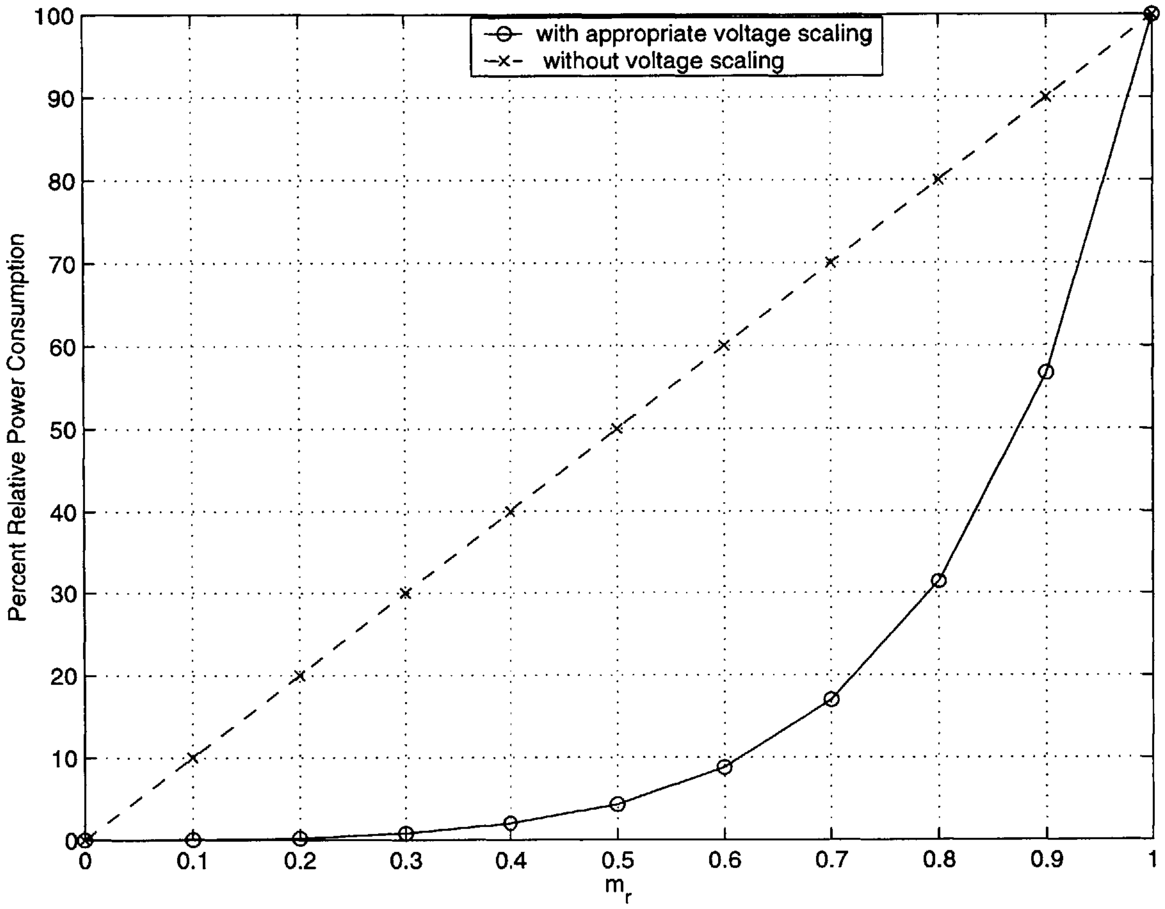

Figure 4-4: Relative percent power consumption as a function of the sampling process mean m_r. A value of $\alpha = 1.3$ was used in this simulation. [Chandrakasan *et al.* [6] derive a similar relationship where the relative power consumption is plotted as a function of the normalized workload.]

The increase in the delay can be achieved by reducing the supply voltage V_{dd}. The relationship between V_{dd} and the circuit delay T_d is given in [19] and shown in equation (4-5). C_L is the load capacitance, α is the velocity saturation index which can be anywhere between 1 (complete velocity saturation) and 2 (no velocity saturation), β is the gate transconductance, and V_t is the device threshold.

$$T_d = \frac{C_L V_{dd}}{\beta(V_{dd} - V_t)^{\alpha}} \tag{4.4}$$

From equation (4.4), we can derive a normalized delay, T_d/T_d^{ref} where T_d^{ref} is the delay when $V_{dd}/V_t = 6$. The normalized delay is given in equation (4.5) where k represents the ratio of V_{dd} to V_t. Figure 4-5 shows a plot of the delay versus V_{dd}/V_t for different values of α. The delay increases hyperbolically as the supply voltage approaches V_t.

$$\frac{T_d}{T_d^{ref}} = \frac{5^\alpha k}{6(k-1)^\alpha}. \tag{4.5}$$

Equation (4.1) suggests that a reduction in the supply voltage will lead to a quadratic reduction in switching power consumption since the power is proportional to the square of the supply voltage. This assumes however that all other variables in that equation remain constant in particular the switching frequency, f_s. However, f_s represents the number of switching events per unit time and circuits are usually operated at the maximum frequency allowed by their propagation delay. As a result, as the propagation delay increases, the number of switching events per unit time, i.e. the switching frequency f_s, will decrease. The resulting decrease in the switching frequency will lead to further improvement in power consumption since switching power is proportional to f_s. The switching power consumption as a function of $k = V_{dd}/V_t$, normalized to power consumption when $V_{dd}/V_t = 6$, is given in equation (4.6).

$$\frac{P_s}{P_s^{ref}} = \frac{k(k-1)^\alpha}{6(5^\alpha)} \tag{4.6}$$

A plot of normalized switching power as a function of the supply voltage for different values of α is shown in Figure 4-5.

4.1.3 Total Power Savings

As discussed above, randomized sampling results in power savings due to switching activity reduction as well as power reduction resulting from voltage scaling. If the mean of the sampling process is m_r, then the analysis carried out in the previous sections shows that N_i is reduced by a factor of $1/m_r$, f_s is also reduced by a factor of $1/m_r$ since the propagation delay can be increased by $1/m_r$. This increase in propagation delay leads to a reduction in the supply voltage by a factor of V_r which can be obtained using equation (4.4). As a result, the total power reduction is by a factor of V_r/m_r^2. Figure 4-4 shows the normalized power consumption as a function of m_r with and without voltage scaling. It is clear from

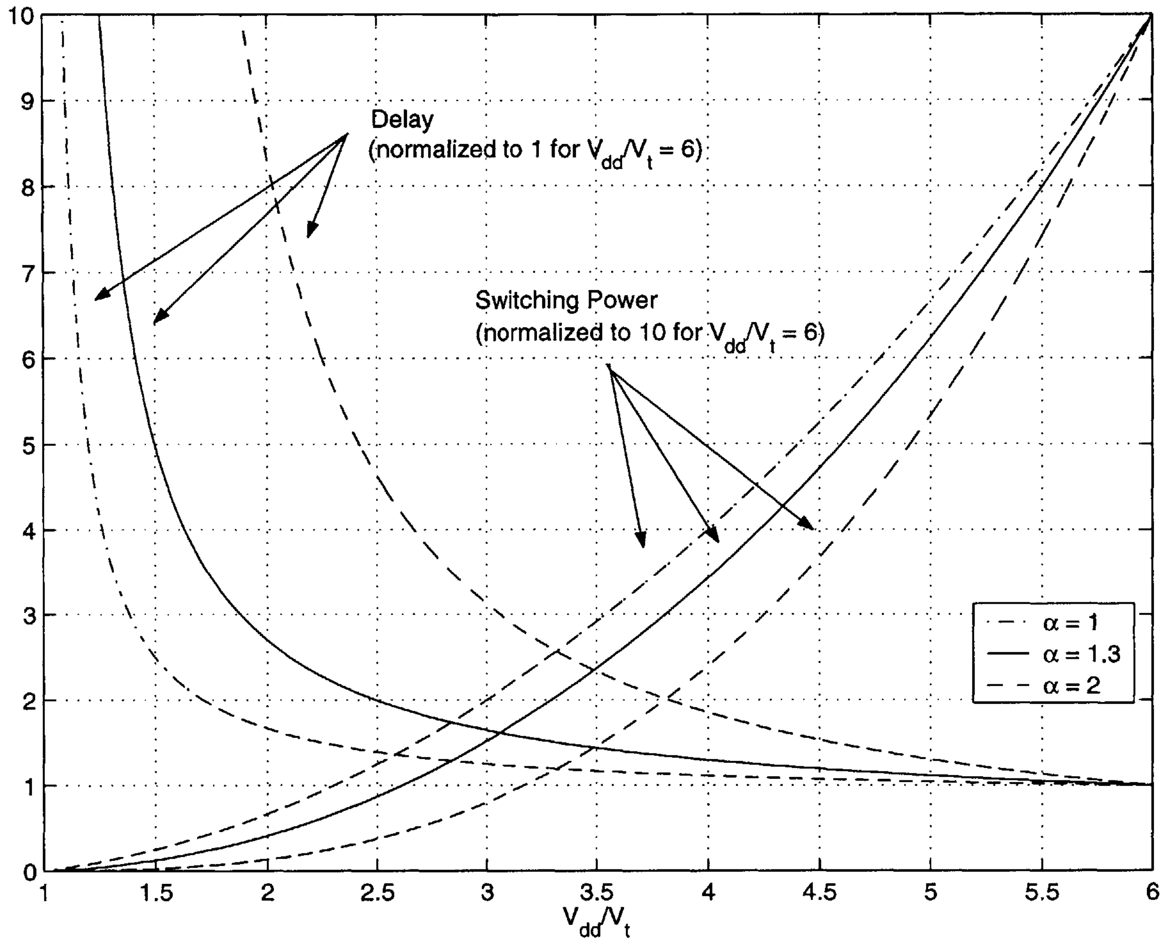

Figure 4-5: Normalized propagation delay and average switching power dissipation for three values of α. The value $\alpha = 1.3$ correspond to the value for 0.25-μm technology [9] while the other two values are the upper and lower bound for α.

this figure that the most savings using randomized sampling occur indirectly through a reduction in the supply voltage.

4.2 Tradeoff between Power and Quality

A simple performance measure is given by total SNR, i.e. the ratio of total output signal power to total output error power. For the IDTRS case, the total SNR, expressed in dB, is given by:

$$\text{SNR (in dB)} = 10\log_{10}(\frac{m_r}{1-m_r}) + 10\log_{10}(\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}). \tag{4.7}$$

The second term in equation (4.7) is independent of the sampling process parameters and as a result, the first term can be used as a measure for gain in SNR as m_r varies. Figure 4-6 gives a plot of normalized power consumption versus gain in SNR for different values of m_r. The figure shows that for power savings greater than 90%, the SNR degrades sharply,

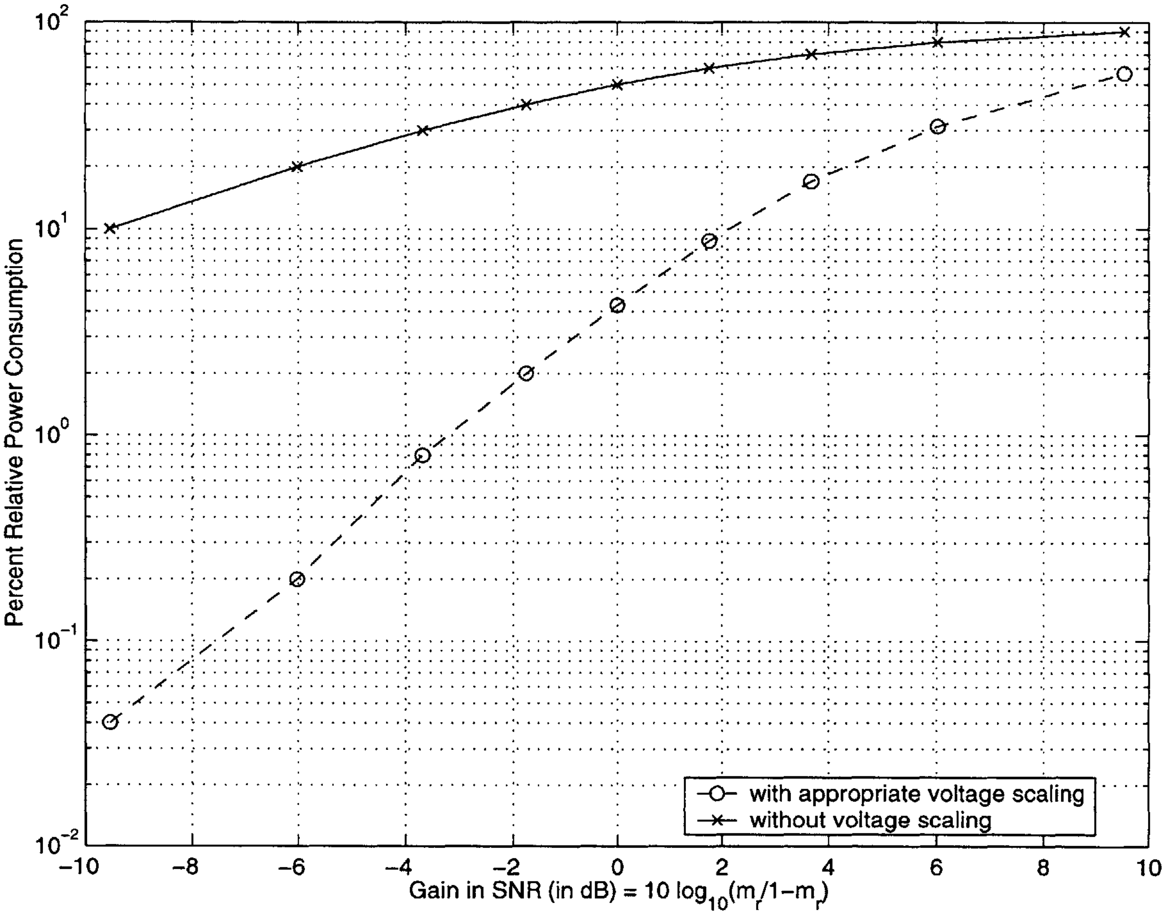

Figure 4-6: Relative percent power consumption as a function of the gain in SNR at the output on a semi-log scale. A value of $\alpha = 1.3$ was used.

however for power savings around 50% the gain in SNR is higher than 5 dB.

4.3 Summary

A low-power formulation for discrete-time randomized sampling was presented in this chapter and two levels of power savings were exploited. Power was first reduced through a lower switching activity due to a smaller number of operations since multiply-by-zero operations can be checked for and skipped. Power was then further reduced by scaling the supply voltage since the lower number of multiplications allows the slowing-down of the processor while maintaining a constant throughput. While the additional savings through lower voltage supply are not specific to randomized sampling but a consequence of reduced computation, the analysis was used to derive the relationship between total savings and the mean of the sampling process and therefore the tradeoff between power savings and SNR, which is specific to randomized sampling, was examined and quantified.

Chapter 5

Hardware Failure Modeling

With the growing prevalence of digital systems in communication networks, it is increasingly important to develop models for hardware failures and to design fault tolerant systems to deal with such errors. Most current fault tolerant algorithms address only single, statistically independent faults [17]. They also tend to be very complex and inefficient. In these cases, efficiency and simplicity are traded for robustness. This chapter presents a different approach where the goal is to maintain a pre-set tolerable signal-to-noise ratio in the presence of certain types of hardware failures as opposed to preventing errors. The types of failures considered are first examined, then an analysis of the signal-to-noise ratio resulting from some systems is analyzed. Finally, algorithms that are guaranteed to maintain a desired SNR under a given probability of failure are developed.

5.1 Hardware Failure Model

Consider direct and transposed direct form implementations of an FIR filter as shown in Figures 5-1 and 5-2 where we have explicitly drawn the multipliers, adders, and delay elements. The delay elements can be thought of as memory registers, the filter coefficients are also stored in memory or specialized registers, and it is assumed that the output of the multipliers are stored in registers where the adders can access those values. At any given time point, n, and for a given coefficient, $h[i]$, the following three types of failures generate the same error and are therefore equivalent from a distortion point of view:

1. The value of the coefficient is read as a zero.

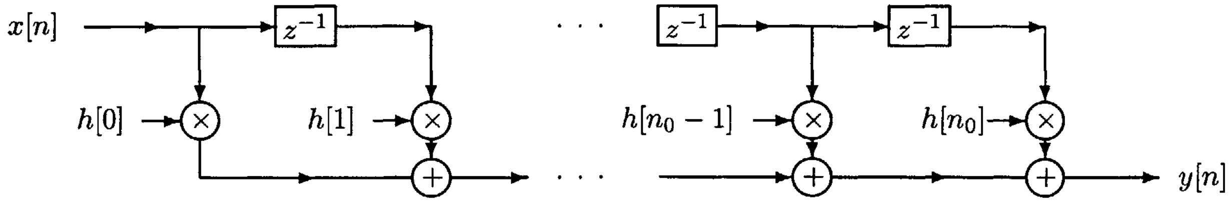

Figure 5-1: Direct Form Implementation of an FIR Filter.

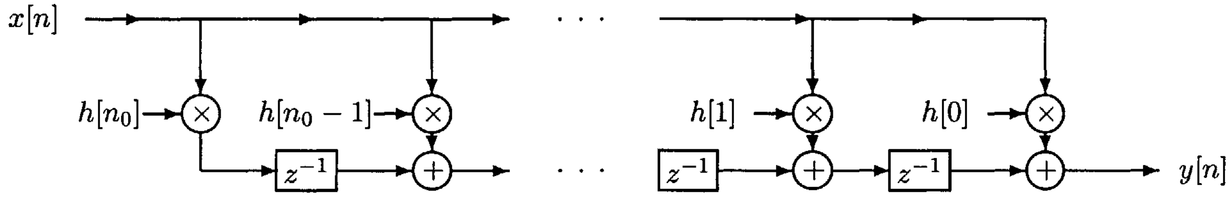

Figure 5-2: Transposed Direct Form Implementation of an FIR Filter.

2. The output of the multiplier associated with the coefficient is reset to zero.

3. An addition is skipped by not using the value resulting from the multiplier operation associated with the coefficient.

In addition, assume that, for each coefficient, $h[i]$, the probability of an error occurring, which results from one of the above modes of failures, is $(1 - m_r)$ and is independent of the occurrence of other similar errors associated with other coefficients. The resulting faulty process is, in this case, equivalent to iterative randomized sampling of the FIR impulse response as described in Chapter 2. In particular, under these circumstances and after proper scaling, the distortion introduced consists of additive zero-mean white noise at the output with variance $\frac{1-m_r}{m_r}R_{xx}[0]R_{hh}[0]$, i.e. Figures 5-3 and 5-4 are equivalent up to second-order statistics.

5.2 Bandlimited Input

Since the distortion resulting from the hardware failures described in the previous section consists of additive white noise, one can use simple techniques to improve the signal-to-

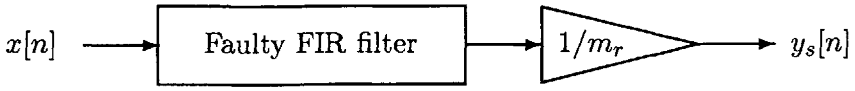

Figure 5-3: FIR Filtering using a Faulty FIR Filter with Output Scaling.

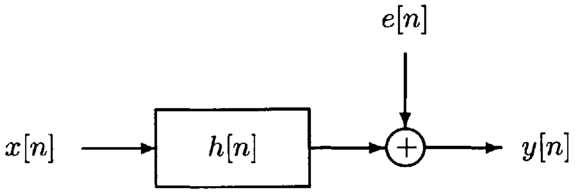

Figure 5-4: Additive Noise Model for the Faulty FIR Filter with Proper Re-scaling Shown in Figure 5-3. $e[n]$ is a zero-mean wide-sense-stationary white noise with variance $\frac{1-m_r}{m_r}R_{xx}[0]R_{hh}[0]$.

noise ratio at the output. In particular, if the input signal is bandlimited to $\frac{\pi}{N}$, then a more relevant measure is the in-band signal-to-noise ratio since out-of-band noise can be filtered out by using an appropriate low-pass filter. In this case, the output will also be bandlimited to $\frac{\pi}{N}$, and as a result the in-band-SNR, SNR$_{in}$, is given by:

$$\text{SNR}_{in} = \frac{\text{signal power}}{\text{noise power in signal band}} = N\frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]} \tag{5.1}$$

$$\text{SNR}_{in}\ (\text{in dB}) = 10\log_{10}(\frac{Nm_r}{1-m_r}) + 10\log_{10}(\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}) \tag{5.2}$$

Note that, as in the case of the SNR computed in Chapter 4, the second term on the right hand side of equation (5.2) only depends on the input and desired filter while the first term incorporates the effect of the probability of failure, $(1-m_r)$, as well as N on the SNR . As a result, this first term can be regarded as a gain factor that incorporates all the effect on SNR due to the probability of error and N. The relationship between the gain in SNR, N, and m_r is shown in Figures 5-5, 5-6, and 5-7 which display how N varies with m_r at constant gain in SNR levels, how the gain in SNR varies with increased m_r and how it varies with increased N, respectively.

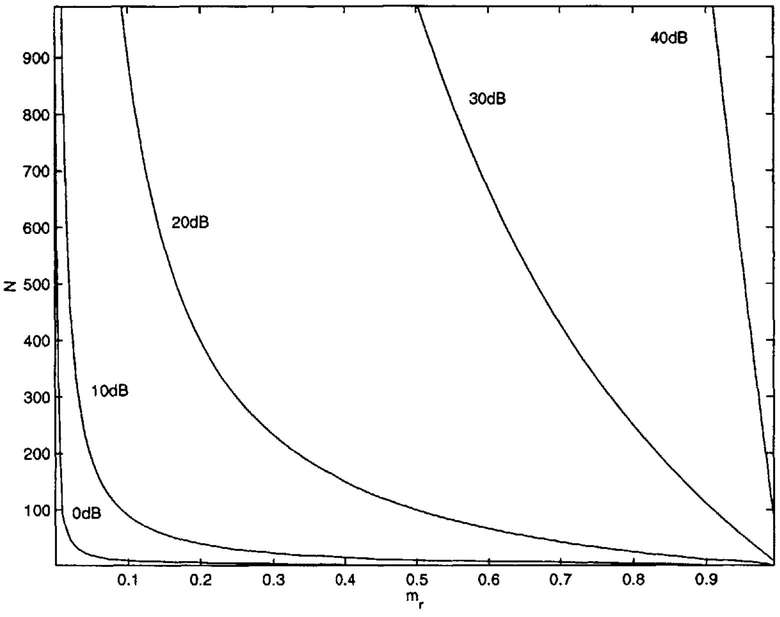

Figure 5-5: The Value of N Needed to Maintain a Given Gain in SNR. The probability of a hardware failure is given by $(1 - m_r)$.

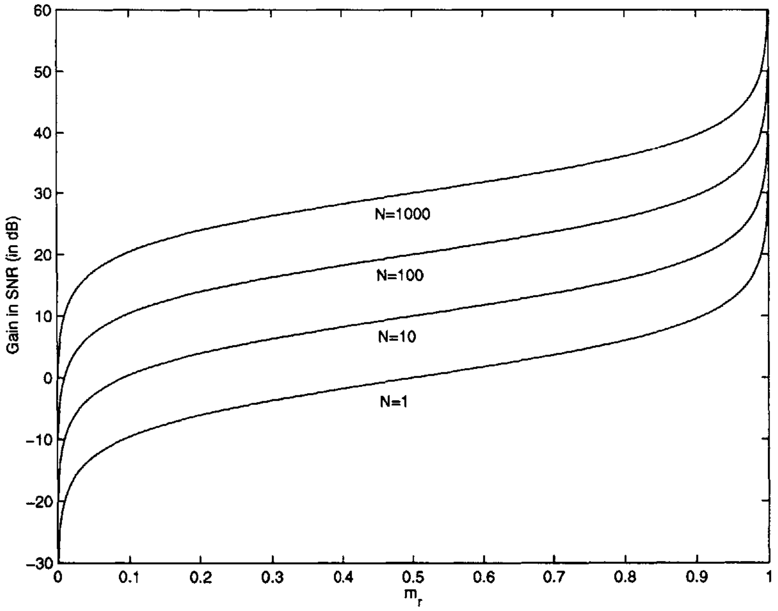

Figure 5-6: Gain in SNR as a Function of m_r for Different Values of N. The probability of a hardware failure is given by $(1 - m_r)$.

Bandlimited inputs can occur naturally or be obtained by up-sampling the input and filtering it at a higher rate. Figures 5-8 and 5-10 show the original filter implementation and an equivalent implementation that uses the up-sampled version of the input and filters it at a higher rate. The filter used in the up-sampled version, $h_{up}[n]$, is obtained by up-sampling the original filter, $h[n]$, as shown in Figure 5-9. It should be noted that both $x[n]$ and $h[n]$ are fully up-sampled i.e. zeros are inserted and those values are properly interpolated. The up-sampling is such that the up-sampled signals have the same energy as the original signals.

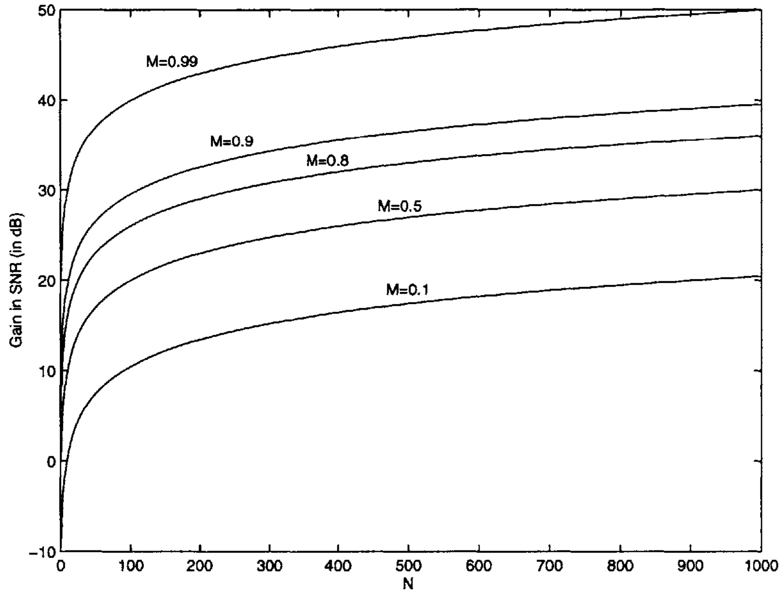

Figure 5-7: Gain in SNR as a function of N for different values of m_r.

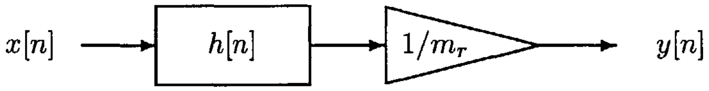

Figure 5-8: Original System with Scaling.

Based on equation (5.2), the resulting system leads to a gain of N in SNR. Note that prior to the second low-pass filter, LPF2, in Figure 5-10, the SNR as well as the probability of failure are the same as for the unmodified case. The averaging resulting from the second low-pass filter leads to an increase in SNR because it eliminates all high frequency components

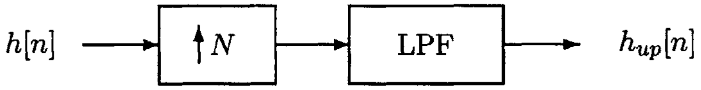

Figure 5-9: Derivation of $h_{up}[n]$. LPF is a low-pass filter with cutoff $\frac{\pi}{N}$ and gain N.

of the error, therefore reducing the error variance. Averaging however does not distort the signal since the prior oversampling ensures that all the spectral information in the signal occurs at low enough frequency. This approach, and the intuition behind it, is similar to oversampled analog-to-digital conversion. In addition, Figure 5-5 can be used as a guide to deciding on the amount of up-sampling needed to maintain a given SNR given a certain probability of failure. For example up-sampling by a factor of 10 will be needed to maintain a gain in SNR of 20dB with a 1% probability of failing.

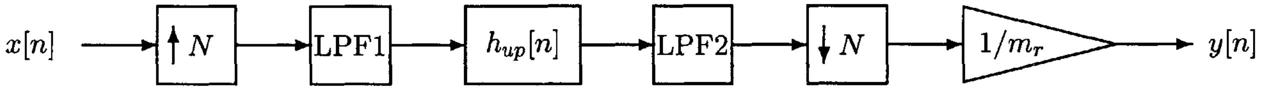

Figure 5-10: Up-sampled Realization of Figure 5-8. LPF1 is a low-pass filter with cutoff $\frac{\pi}{N}$ and gain N while LPF2 is a low-pass filter with cutoff $\frac{\pi}{N}$ and unit gain. It is assumed that only $h_{up}[n]$ is implemented on the faulty hardware.

5.3 Parallel Filters

The signal-to-noise ratio can also be improved by using multiple faulty filters in parallel and averaging the outputs from each filter. Such topology can be particularly desirable in a sensor context where several sensors are deployed to measure the same signal by processing it through an FIR filter. The output then consists of an average of the individual outputs from each sensor. Figure 5-11 displays this topology where each of the sensors has components that can fail, with probability $(1 - m_r)$, in a manner similar to Section 5.2. In addition, a scaling factor is included to account for the loss in energy due to the failing process. Figure 5-12 shows an equivalent network generated using the IDTRS model. In this figure, each sensor is modeled as the original FIR filter, $h[n]$, with white noise added to its output. Each of the noise processes, $w_i[n]$, is zero-mean wide-sense-stationary white uncorrelated with the input with variance $\frac{1-m_r}{m_r} R_{xx}[0] R_{hh}[0]$. Using Figure 5-12, we can write $y_r[n]$ as:

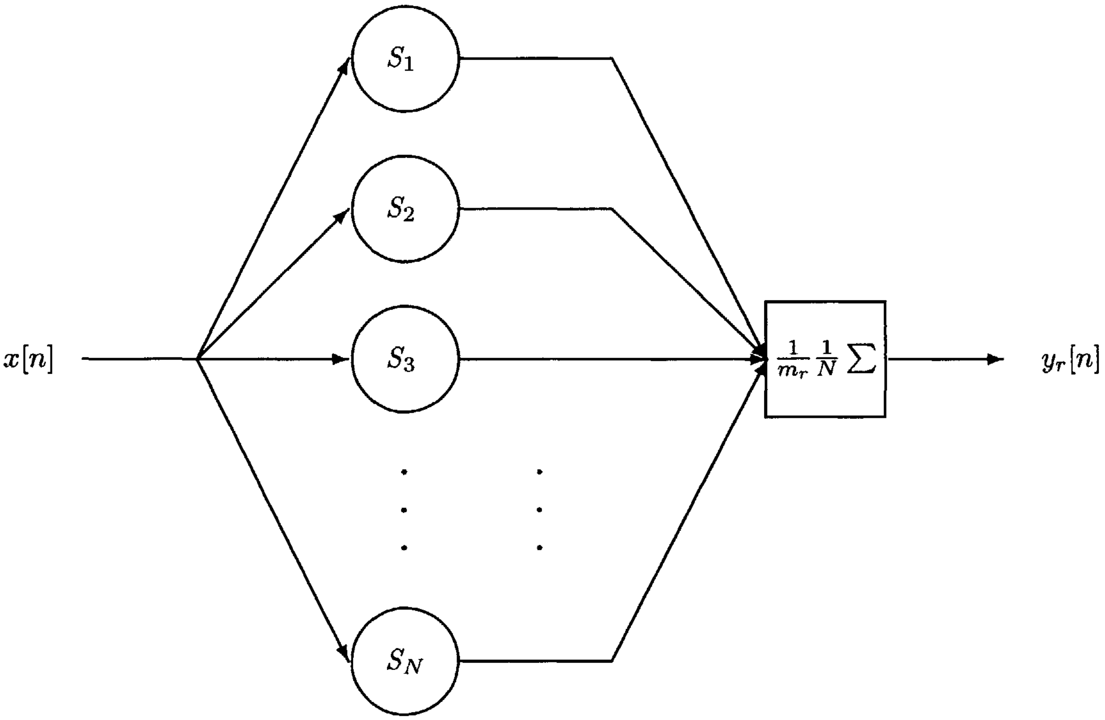

Figure 5-11: Sensor Network.

$$y_r[n] = y[n] + e[n] \tag{5.3}$$

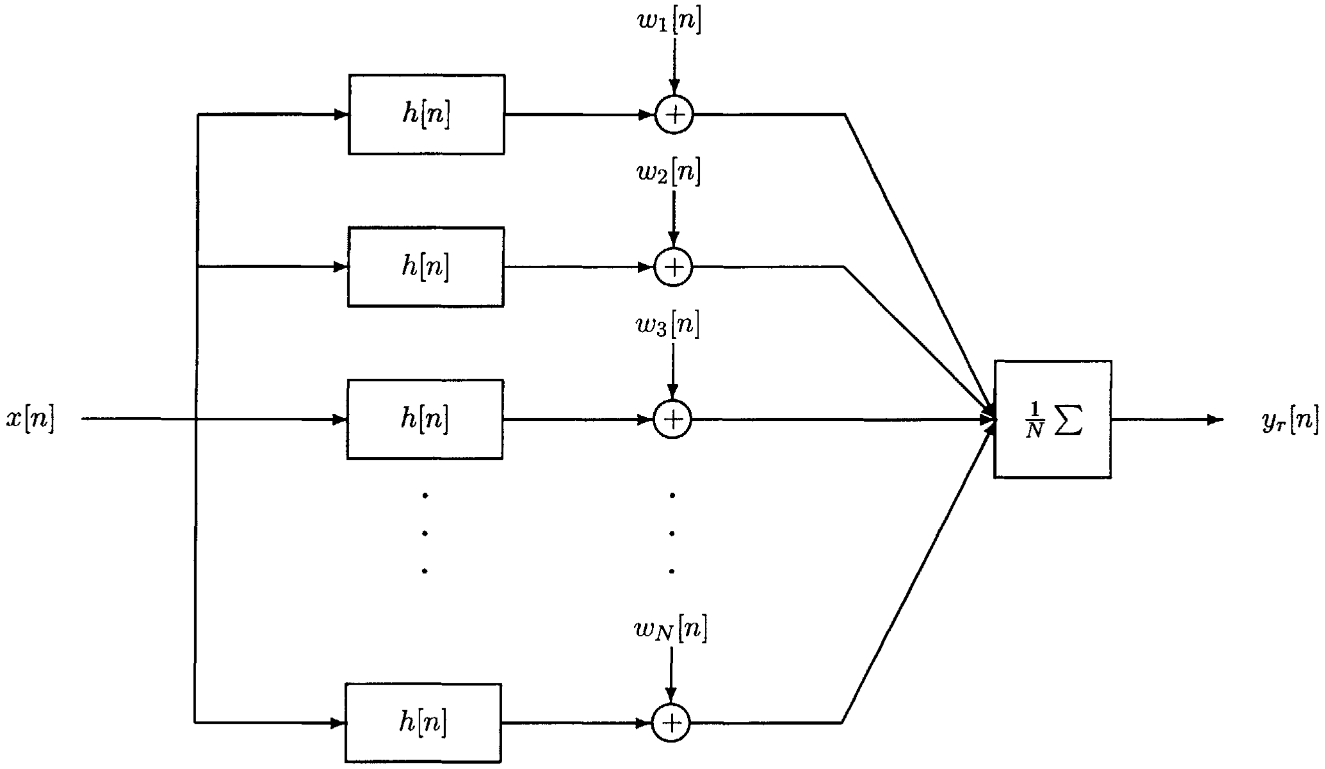

Figure 5-12: Sensor Network Model.

where $y[n] = x[n] * h[n]$ and $e[n]$ is a zero-mean wide-sense-stationary white noise process with variance $N\frac{1-m_r}{m_r}R_{xx}[0]R_{hh}[0]$. The SNR at the output is therefore:

$$\text{SNR} = \frac{\text{signal power}}{\text{noise power}} = N\frac{m_r}{1-m_r}\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]} \tag{5.4}$$

$$\text{SNR (in dB)} = 10\log_{10}(\frac{Nm_r}{1-m_r}) + 10\log_{10}(\frac{R_{yy}[0]}{R_{xx}[0]R_{hh}[0]}) \tag{5.5}$$

Equation (5.5) is the same as equation (5.2) and therefore Figures 5-5, 5-6, and 5-7 display the tradeoff between gain in SNR and the error probability as well as N where now N refers to the number of filters or sensors used in parallel. For example, to maintain a gain of 30dB in SNR, 15 sensors should be used in parallel if the probability of error is 1%.

5.4 Summary

This chapter used discrete-time randomized sampling as a framework to model independent random failures of adders, multipliers and register retrieving operations in hardware implementations of direct-form and transposed direct-form FIR filters. The resulting additive noise model lead to the formulation of two algorithms to increase the signal-to-noise ratio at the output. The first consisted of up-sampling the input by a factor of N and filtering it at a higher rate while the second used N processors in parallel and averaged their output. Both algorithms lead to a gain of N in SNR and graphs that illustrate the tradeoffs between gain in SNR, N and the probability of failure were presented.

Chapter 6

Conclusion

6.1 Thesis Contributions

This thesis defined a framework, discrete-time randomized sampling, and explored applications in three areas: approximate filtering, low-power design, and hardware failure modeling. The thesis contributions are two-fold:

1. A formalized discrete-time randomized sampling theory.

   and

2. Randomized sampling algorithms spanning various signal processing areas.

In this section, we summarize the analysis carried out in previous chapters and outline the thesis results.

Discrete-time randomized sampling was defined as a process which randomly sets some of the sample points of a discrete-time signal or filter impulse response to zero. We specifically derived three sampling techniques: *discrete-time randomized sampling of the input* (*DTRS of the input*), *discrete-time randomized sampling of the impulse response* (*DTRS of the impulse response*), and *iterative discrete-time randomized sampling of the impulse response* (*IDTRS*). In DTRS of the input (the impulse response), a random subset of the input signal (impulse response) coefficients is set to zero while in IDTRS the filter impulse response is randomly sampled at each iteration of the convolution sum. After deriving linear noise models for each sampling scheme where the system was expressed as the original output added to an error term uncorrelated with the signal, error characteristics in the case of white sampling were analyzed for each scheme. Semi-white sampling, where the two-dimensional autocorrelation function of the sampling process is white in only one dimension and correlated in the other, was also explored in the case of iterative randomized sampling and error analysis was performed.

In order to assess the quality of each sampling procedure, the in-band SNR was analyzed and conditions where IDTRS led to the best performance were derived. Specifically, it was shown that if the filter (the input power) is bandlimited to ω_0 and the ratio of the input power (filter power) within the filter (input) band to the total power in the input (filter) is greater than ω_0/π, then IDTRS always leads to the highest in-band SNR. We demonstrated that error characteristics (error variance and in-band SNR) could also be improved by imposing more structure to the sampling process leading to the notion of trading-off sampling structure for better error characteristics. For example, if the input process is colored, the error variance in the iterative randomized sampling scheme can be improved by using one-lag correlated sampling at each convolution step. Alternatively, higher in-band SNR can be achieved by using a sampling process with one-lag correlation across output samples.

Chapter 3 explored the potential of randomized sampling as an approximation technique by comparing approximations based on randomized sampling to LTI approximations in the context of constrained linear minimum mean-square estimation. We first proved that if the input to the estimator is white, then the best length M linear estimator, in the mean-square sense, is the truncated Wiener filter. As a result, conditions were derived under which randomized sampling performed better than the best constrained length M estimator. It was shown in that case that if the energy of the Wiener filter was sufficiently spread out in time, then IDTRS of the original estimator lead to a better mean-square estimate than the optimal fixed-length LTI mean-square estimator. Specifically, if the ratio of the energy in the out-of-window region of the Wiener filter to the energy in the windowed impulse response is greater than $(1 - m_r)/(2m_r - 1)$, then IDTRS always outperforms LTI approximations. These results could also be incorporated for the case where the data is not white. In particular, the Wiener filter could be, in that case, decomposed into a whitening filter and an estimator operating on the whitened data, randomized sampling can then be used to approximate the estimator part of the system.

A low-power formulation for randomized sampling was presented in Chapter 4 where, by checking for multiply-by-zero operations, the switching activity was reduced. Additional power savings were exploited by noting that throughput could be maintained by slowing down the processor and therefore reducing the voltage supply. We showed that power savings as high as 70% could be obtained if the impulse response is sampled with a mean of 0.8. The tradeoff between quality and power in the case of white iterative randomized sampling was further analyzed and a relationship between percent relative power consumption and gain in SNR was obtained. It was shown that for power savings of the order of 50%, the gain in SNR was higher than 5dB.

DTRS was also used for modeling of a class of hardware failures. In Chapter 5, we showed that three types of random failures of adders, multipliers, and register-retrieving operations of direct and transposed-direct form hardware implementations of FIR filters could be modeled as IDTRS. As a result, the linear models derived in Chapter 2 were used to develop algorithms for maintaining a desired SNR given a probability of hardware failure. An approach involved first up-sampling the input by a factor of N and filtering it at a higher rate using the faulty hardware then down-sampling the output while another approach averaged the outputs to a collection of N identical faulty FIR filters. Graphs showing the tradeoff between gain in SNR, N, and the probability of failure were also presented. For example, it was shown that up-sampling the input by a factor of 10 would be needed to maintain a gain in SNR of 20dB under a 1% probability of failure.

6.2 Future Directions

The framework defined by this thesis and the analysis and applications explored revealed some of the potential as well as limitations of discrete-time randomized sampling. However, a number of issues and applications which may lead to interesting results and innovative algorithms still need to be investigated. In this section, we suggest some directions for further work.

First, the main focus of this thesis was white sampling processes and while some semi-white processes were explored, a thorough investigation of correlated sampling should be carried out. We suspect that better error characteristics could be obtained from additional structure in the sampling process through increased correlation. In particular, a potentially promising method is energy-proportional sampling where the probability of setting a coefficient to zero is inversely proportional to the normalized energy of that coefficient. In this case, the sampling process is no longer wide-sense-stationary, however it is still simple enough to analyze.

Second, in Chapter 3, we saw that the randomized sampling error was fundamentally different from any LTI filter approximation method since the error is signal-independent in the randomized case whereas it is signal-dependent in the LTI case. There are many applications where a high-variance uncorrelated white error is more desirable than a low-variance correlated colored error. Audio and image processing are examples of such applications. Since an analytical perceptual metric for audio and speech processing is hard to obtain, hearing experiments should be performed to compare approximation techniques using randomized sampling to other optimal LTI approximation techniques. We believe that randomized sampling may well lead to a perceptually higher quality output than other LTI approximation techniques. We suggest starting with applications in analog-to-digital speech pre-processing where the pre-emphasis filter is approximated. Similarly, approximate image processing applications need to be explored.

Third, in the domain of low-power filtering, simulations should be carried out to measure the actual power savings versus quality distortion. Furthermore, Chapter 4 of this thesis evaluated the tradeoff between power consumption and output quality only for the case of white sampling. This tradeoff should be investigated for correlated sampling schemes since improved quality is expected in these cases.

Fourth, in the domain of hardware failure modeling, more algorithms should be explored to improve the quality of the output given a hardware failure probability. Specifically, efforts should be invested in trying to identify hardware constellations where while each piece of the constellation fails according to the white iterative randomized sampling model, the interconnections are such that the overall system exhibits a composite failing behavior characteristic of correlated sampling and as a result, achieves noise shaping and higher SNR.

Finally, we suggest exploring *adaptive discrete-time randomized sampling* where the sampling process mean and autocorrelation function are adjusted according to some target quality performance and complexity level subject to a pre-defined metric. In this case, depending on the input signal statistics, the sampling process varies from white and constant mean to correlated and varying mean in order to maintain a fixed output quality. In addition, for any given time point, this scheme guarantees the use of the simplest and most efficient (lowest mean) sampling process to achieve a target quality performance.

Appendix A

Derivations for White Sampling

A.1 $m_r y[n]$ and $e_1[n]$ are uncorrelated

$$\begin{aligned} y_{s1}[n] &= h[n] * (x[n]r_s[n]) && \text{(A.1)} \\ &= h[n] * ((m_r + \hat{r}_s[n])x[n]) && \text{(A.2)} \\ &= m_r y[n] + e_1[n] && \text{(A.3)} \end{aligned}$$

Proof that $m_r y[n]$ and $e_1[n]$ are uncorrelated:

$$
\begin{aligned}
E\{m_r y[n] e_1[n]\} &= m_r E\{(x[n] * h[n])((\hat{r}_s[n]x[n]) * h[n])\} && \text{(A.4)}\\
&= m_r E\{ \sum_{k=-\infty}^{+\infty} x[k]h[n-k] && \text{(A.5)}\\
& \sum_{l=-\infty}^{+\infty} \hat{r}_s[l]x[l]h[n-l]\} && \text{(A.6)}\\
&= m_r E\{ \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} (x[k]h[n-k]) && \text{(A.7)}\\
& (\hat{r}_s[l]x[l]h[n-l])\} && \text{(A.8)}\\
&= m_r \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\{x[k]x[l]h[n-k]h[n-l]\hat{r}_s[l]\} && \text{(A.9)}\\
&= m_r \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\{\hat{r}_s[l]\} && \text{(A.10)}\\
& E\{x[k]x[l]h[n-k]h[n-l]\} && \text{(A.11)}\\
&= 0 && \text{(A.12)}
\end{aligned}
$$

Also:

$$
\begin{aligned}
E\{m_r y[n]\}E\{e_1[n]\} &= E\{m_r y[n]\}E\{(\hat{r}_s[n]x[n]) * h[n]\} && \text{(A.13)}\\
&= E\{m_r y[n]\}E\{ \sum_{k=-\infty}^{+\infty} \hat{r}_s[k]x[k]h[n-k]\} && \text{(A.14)}\\
&= E\{m_r y[n]\} \sum_{k=-\infty}^{+\infty} E\{\hat{r}_s[k]x[k]h[n-k]\} && \text{(A.15)}\\
&= E\{m_r y[n]\} \sum_{k=-\infty}^{+\infty} E\{\hat{r}_s[k]\}E\{x[k]h[n-k]\} && \text{(A.16)}\\
&= 0 && \text{(A.17)}
\end{aligned}
$$

Therefore:

$$
E\{m_r y[n] e_1[n]\} = E\{m_r y[n]\}E\{e_1[n]\} \qquad \text{(A.18)}
$$

A.2 The autocorrelation of the error: $R_{e_1e_1}[m]$

$$\begin{aligned}
R_{e_1e_1}[m] &= E\{e_1[n]e_1[n+m]\} && \text{(A.19)}\\
&= E\{\sum_{k=-\infty}^{+\infty} \hat{r}_s[k]x[k]h[n-k] \sum_{l=-\infty}^{+\infty} \hat{r}_s[l]x[l]h[n+m-l]\} && \text{(A.20)}\\
&= E\{\sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} \hat{r}_s[k]\hat{r}_s[l]x[k]x[l]h[n-k]h[n+m-l]\} && \text{(A.21)}\\
&= \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\{\hat{r}_s[k]\hat{r}_s[l]\}E\{x[k]x[l]\}h[n-k]h[n+m-l]\} && \text{(A.22)}\\
&= \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} R_{\hat{r}_s\hat{r}_s}[k-l]R_{xx}[k-l]h[n-k]h[n+m-l]\} && \text{(A.23)}
\end{aligned}$$

Since:

$$R_{\hat{r}_s\hat{r}_s}[m] = \sigma_r^2\delta[m] = m_r(1-m_r)\delta[m] \qquad \text{(A.24)}$$

Then:

$$\begin{aligned}
R_{e_1e_1}[m] &= \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} m_r(1-m_r)\delta[k-l]R_{xx}[k-l]h[n-k]h[n+m-l] && \text{(A.25)}\\
&= \sum_{k=-\infty}^{+\infty} m_r(1-m_r)\delta[0]R_{xx}[0]h[n-k]h[n+m-k] && \text{(A.26)}\\
&= m_r(1-m_r)R_{xx}[0]R_{hh}[m] && \text{(A.27)}
\end{aligned}$$

A.3 $m_ry[n]$ and $e_2[n]$ uncorrelated

$$\begin{aligned}
E\{m_ry[n]e_2[n]\} &= m_rE\{(x[n]*h[n])(x[n]*(\hat{r}_s[n]h[n]))\} && \text{(A.28)}\\
&= m_rE\{\sum_{k=-\infty}^{+\infty} x[k]h[n-k] \sum_{l=-\infty}^{+\infty} x[l]\hat{r}_s[n-l]h[n-l]\} && \text{(A.29)}\\
&= m_r \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\{x[k]x[l]h[n-k]h[n-l]\hat{r}_s[n-l]\} && \text{(A.30)}\\
&= m_r \sum_{k=-\infty}^{+\infty} \sum_{l=-\infty}^{+\infty} E\{x[k]x[l]h[n-k]h[n-l]\}E\{\hat{r}_s[n-l]\} && \text{(A.31)}\\
&= 0 && \text{(A.32)}
\end{aligned}$$

Also:

$$\begin{aligned} E\{m_r y[n]\}E\{e_2[n]\} &= E\{m_r y[n]\}E\{\sum_{k=-\infty}^{+\infty} x[k]\hat{r}_s[n-k]h[n-k]\} && \text{(A.33)} \\ &= E\{m_r y[n]\}\sum_{k=-\infty}^{+\infty} E\{x[k]h[n-k]\}E\{\hat{r}_s[n-k]\} && \text{(A.34)} \\ &= 0 && \text{(A.35)} \end{aligned}$$

Therefore:

$$E\{m_r y[n] e_2[n]\} = E\{m_r y[n]\}E\{e_2[n]\} \qquad \text{(A.36)}$$

A.4 Derivation of $R_{e_2e_2}[m]$

$$\begin{aligned} R_{e_2e_2}[m] &= E\{e_2[n]e_2[n+m]\} && \text{(A.37)} \\ &= E\{\sum_{k=-\infty}^{+\infty} \hat{r}_s[k]h[k]x[n-k] \sum_{l=-\infty}^{+\infty} \hat{r}_s[l]h[l]x[n+m-l]\} && \text{(A.38)} \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} E\{\hat{r}_s[k]\hat{r}_s[l]h[k]h[l]x[n-k]x[n+m-l]\} && \text{(A.39)} \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]E\{\hat{r}_s[k]\hat{r}_s[l]\}E\{x[n-k]x[n+m-l]\} && \text{(A.40)} \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{\hat{r}_s\hat{r}_s}[k-l]R_{xx}[-k-m+l] && \text{(A.41)} \end{aligned}$$

Since:

$$R_{\hat{r}_s\hat{r}_s}[m] = m_r(1-m_r)\delta[m] \qquad \text{(A.42)}$$

Then:

$$\begin{aligned}
R_{e_2e_2}[m] &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]m_r(1-m_r)\delta[k-l]R_{xx}[l-m-k] && \text{(A.43)}\\
&= \sum_{k=-\infty}^{+\infty} h[k]h[k]m_r(1-m_r)\delta[0]R_{xx}[-m] && \text{(A.44)}\\
&= m_r(1-m_r)R_{xx}[m]\sum_{k=-\infty}^{+\infty} h[k]h[k] && \text{(A.45)}\\
&= m_r(1-m_r)R_{hh}[0]R_{xx}[m] && \text{(A.46)}
\end{aligned}$$

A.5 $m_r y[n]$ and $e_3[n]$ are uncorrelated

$$\begin{aligned}
E\{m_r y[n]e_3[n]\} &= m_r E\{(x[n]*h[n])(x[n]*(\hat{r}_s[n;k]h[n]))\} && \text{(A.47)}\\
&= m_r E\{\sum_{k=-\infty}^{+\infty} x[k]h[n-k]\sum_{l=-\infty}^{+\infty} h[l]\hat{r}_s[n;l]x[n-l]\} && \text{(A.48)}\\
&= m_r \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} E\{x[k]x[n-l]h[n-k]h[l]\hat{r}_s[n;l]\} && \text{(A.49)}\\
&= m_r \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} E\{x[k]x[n-l]h[n-k]h[l]\}E\{\hat{r}_s[n;l]\} && \text{(A.50)}\\
&= 0 && \text{(A.51)}
\end{aligned}$$

Also:

$$\begin{aligned}
E\{m_r y[n]\}E\{e_3[n]\} &= E\{m_r y[n]\}E\{\sum_{k=-\infty}^{+\infty} h[k]\hat{r}_s[n;k]x[n-k]\} && \text{(A.52)}\\
&= E\{m_r y[n]\}\sum_{k=-\infty}^{+\infty} E\{h[k]\hat{r}_s[n;k]x[n-k]\} && \text{(A.53)}\\
&= E\{m_r y[n]\}\sum_{k=-\infty}^{+\infty} E\{h[k]x[n-k]\}E\{\hat{r}_s[n;k]\} && \text{(A.54)}\\
&= 0 && \text{(A.55)}
\end{aligned}$$

Therefore:

$$E\{m_r y[n]e_3[n]\} = E\{m_r y[n]\}E\{e_3[n]\} \qquad \text{(A.56)}$$

A.6 The Autocorrelation of $e_3[n]$: $R_{e_3e_3}[m]$

$$
\begin{aligned}
R_{e_3e_3}[m] &= E\{e_3[n]e_3[n+m]\} \qquad (A.57)\\
&= E\{\sum_{k=-\infty}^{+\infty} h[k]\hat{r}_s[n;k]x[n-k] \sum_{l=-\infty}^{+\infty} h[l]\hat{r}_s[n+m;l]x[n+m-l]\} \qquad (A.58)\\
&= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} E\{h[k]h[l]\hat{r}_s[n;k]\hat{r}_s[n+m;l]x[n-k]x[n+m-l]\} \qquad (A.59)\\
&= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]E\{\hat{r}_s[n;k]\hat{r}_s[n+m;l]\}E\{x[n-k]x[n+m-l]\} \qquad (A.60)\\
&= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{\hat{r}_s\hat{r}_s}[m;l-k]R_{xx}[m-l+k] \qquad (A.61)
\end{aligned}
$$

Since:

$$R_{\hat{r}_s\hat{r}_s}[n;k] = \sigma^2\delta[n]\delta[k] = m_r(1-m_r)\delta[n]\delta[k] \qquad (A.62)$$

Then:

$$
\begin{aligned}
R_{e_3e_3}[m] &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]m_r(1-m_r)\delta[m]\delta[l-k]R_{xx}[m-l+k] \qquad (A.63)\\
&= \sum_{k=-\infty}^{+\infty} h[k]h[k]m_r(1-m_r)\delta[m]\delta[0]R_{xx}[m] \qquad (A.64)\\
&= m_r(1-m_r)R_{hh}[0]R_{xx}[0]\delta[m] \qquad (A.65)
\end{aligned}
$$

Appendix B

Derivations for Non-White Sampling

B.1 $R_{e_3e_3}[m]$ in the case of correlated sampling at each convolution step

Using derivations in Appendix A we have:

$$\begin{aligned} R_{e_3e_3}[m] &= E\{e_3[n]e_3[n+m]\} && \text{(B.1)} \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{\hat{r}_s\hat{r}_s}[m;l-k]R_{xx}[m-l+k] && \text{(B.2)} \end{aligned}$$

Since in this case:

$$R_{\hat{r}_s\hat{r}_s}[m;k] = (1-m_r)m_r\delta[m]R_{gg}[k] \tag{B.3}$$

Then:

$$\begin{aligned} R_{e_3e_3}[m] &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l](1-m_r)m_r\delta[m]R_{gg}[l-k]R_{xx}[m-l+k] && \text{(B.4)} \\ &= m_r(1-m_r)\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{xx}[k-l]R_{gg}[l-k]\delta[m] && \text{(B.5)} \end{aligned}$$

In addition, if:

$$R_{gg}[m] = \delta[m] - \alpha\delta[m-1] - \alpha\delta[m+1] \tag{B.6}$$

Then:

$$R_{e_3e_3}[m] = m_r(1-m_r)\sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{xx}[k-l](\delta[l-k] - \alpha\delta[l-k-1] - \alpha\delta[l-k+1])\delta[m] \tag{B.7}$$

$$R_{e_3e_3}[m] = m_r(1-m_r)\delta[m]\sum_{k=-\infty}^{+\infty}(h^2[k]R_{xx}[0] - \alpha h[k]h[k+1]R_{xx}[-1] - \alpha h[k]h[k-1]R_{xx}[1]) \tag{B.8}$$

$$R_{e_3e_3}[m] = m_r(1-m_r)\delta[m](R_{hh}[0]R_{xx}[0] - \alpha R_{xx}[-1]R_{hh}[1] - \alpha R_{xx}[1]R_{hh}[-1]) \tag{B.9}$$

Note that:

$$\begin{aligned} R_{xx}[-1] &= R_{xx}[1] && \text{(B.10)} \\ R_{hh}[-1] &= R_{hh}[1] && \text{(B.11)} \end{aligned}$$

Therefore:

$$\begin{aligned} R_{e_3e_3}[m] &= m_r(1-m_r)\delta[m](R_{hh}[0]R_{xx}[0] - \alpha R_{xx}[1]R_{hh}[1] - \alpha R_{xx}[1]R_{hh}[1]) && \text{(B.12)} \\ &= m_r(1-m_r)(R_{hh}[0]R_{xx}[0] - 2\alpha R_{hh}[1]R_{xx}[1])\delta[m] && \text{(B.13)} \end{aligned}$$

B.2 $R_{e_3e_3}[m]$ in the case of a sampling process correlated across output samples

In this case:

$$R_{\hat{r}_s\hat{r}_s}[m;k] = m_r(1-m_r)R_{vv}[m]\delta[k] \tag{B.14}$$

Therefore:

$$\begin{aligned} R_{e_3e_3}[m] &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{\hat{r}_s\hat{r}_s}[m;l-k]R_{xx}[m-l+k] && \text{(B.15)} \\ &= \sum_{k=-\infty}^{+\infty}\sum_{l=-\infty}^{+\infty} h[k]h[l]R_{xx}[m-l+k]m_r(1-m_r)R_{vv}[m]\delta[l-k] && \text{(B.16)} \\ &= \sum_{k=-\infty}^{+\infty} h^2[k]m_r(1-m_r)R_{xx}[m]R_{vv}[m] && \text{(B.17)} \end{aligned}$$

In addition, if:

$$R_{vv}[m] = \delta[m] - \alpha\delta[m-1] - \alpha\delta[m+1] \tag{B.18}$$

Then:

$$R_{e_3e_3}[m] = m_r(1-m_r)R_{hh}[0]R_{xx}[m](\delta[m] - \alpha\delta[m-1] - \alpha\delta[m+1]) \tag{B.19}$$

$$R_{e_3e_3}[m] = m_r(1-m_r)R_{hh}[0](R_{xx}[0]\delta[m] - \alpha R_{xx}[1]\delta[m-1] - \alpha R_{xx}[-1]\delta[m+1]) \tag{B.20}$$

$$R_{e_3e_3}[m] = m_r(1-m_r)R_{hh}[0]R_{xx}[0](\delta[m] - \alpha\frac{R_{xx}[1]}{R_{xx}[0]}\delta[m-1] - \alpha\frac{R_{xx}[1]}{R_{xx}[0]}\delta[m+1]) \tag{B.21}$$

Bibliography

[1] A.V. Oppenheim, R.W. Schafer with John R. Buck. *Discrete-Time Signal Processing.* Prentice Hall, 1999.

[2] I. Bilinskis, A. Mikelsons. *Randomized Signal Processing.* Prentice Hall, 1992.

[3] R. Brodersen, A. Chandrakasan, S. Sheng. "Low-Power Signal Processing Systems," *Workshop on VLSI Signal Processing*, V, 1992., pp3-13.

[4] T.D. Burd, R.W. Brodersen. "Energy Efficient CMOS Microprocessor Design". *Proceedings of the 28th Annual HICSS Conference.* Jan 1995; Vol I, pp. 288-297.

[5] A.P. Chandrakasan, et. al., "Design of Portable Systems," invited tutorial paper presented at the *Custom Integrated Circuits Conference*, May 1994.

[6] A. Chandrakasan, V. Gutnik, T. Xanthopoulos. "Data Driven Signal Processing: An Approach for Energy Efficient Computing", *ISLPED*, 1996, pp347-352.

[7] G.P.M. Egelmeers, P.C.W. Sommen. "A new method for efficient convolution in frequency domain by nonuniform partitioning for adaptive filtering," *IEEE Transactions on Signal Processing*, Volume: 44 12, Dec. 1996, pp3123-3129.

[8] J.B. Evans, B. Liu. "Some Low Power Implementations of DSP Algorithms," *Proceedings of the Int. Conf. on Application Specific Array Processors*, 1992, pp269-276.

[9] R. Gonzalez, B.M. Gordon, M.A. Horowitz. "Supply and Threshold Voltage Scaling for Low Power CMOS". *IEEE Journal of Solid-State Circuits.* Vol. 32, No 8, August 1997.

[10] J. Jiang, C.D. Schmitz, B.A. Scnaufer and W.K. Jenkins. “The Use of Adaptive Fault Tolerance in General Classes of Linear Systems,” *IEEE 39th Midwest symposium on Circuits and Systems*, 1996, pp1135-1138 vol.3.

[11] J.S. Lim. *Two-Dimensional Signal and Image Processing*. Prentice Hall, 1990.

[12] J.T. Ludwig, S.H. Nawab, A. Chandrakasan. “Low power filtering using approximate processing for DSP applications,” *Proceedings of the IEEE Custom Integrated Circuits Conference*, 1995, pp185-188.

[13] J.T. Ludwig. *Low Power Digital Filtering Using Adaptive Approximate Processing*. Doctor of Philosophy Thesis, MIT, 1997.

[14] T.H. Meng, A.C. Hung, E.K. Tsern, B.M. Gordon. “Low-Power Signal Processing System Design for Wireless Applications,” *IEEE Personal Communications*, June 1998, pp20-31.

[15] D. Mlynek. *Design of VLSI Systems*. Web-based Course, Swiss Federal Intitute of Technology, Lausanne. http://c3iwww.epfl.ch/teaching/webcourse/ch07/ch07.html. 1998.

[16] S.H. Nawab, A.V. Oppenheim, A.P. Chandrakasan, J.M. Winograd, and J.T. Ludwig. “Approximate Signal Processing”, *VLSI Signal Processing Journal*, Vol. 15, No. 1-2, January 1997.

[17] V.P. Nelson. “Fault-Tolerant Computing: Fundamental Concepts,” *Computer*, July 1990, pp19-25.

[18] C.J. Pan. “A low-power digital filter for decimation and interpolation using approximated processing”, *Digest of Technical Papers. 43rd ISSCC., IEEE International solid-State Circuits Conference*, 1997, pp102-103,439.

[19] J.M. Rabaey. *Digital Integrated Circuits: A Design Perspective*. Prentice Hall, New Jersey, 1996.

[20] S.R. Wang, P. Siy. “Parallel-decomposition algorithm for discrete computational problems and its application in developing an efficient discrete convolution algorithm,” *Vision, Image and Signal Processing, IEE Proceedings*, Volume: 142 1, Feb. 1995, pp40-46.