Chapter 2

Sampling

2.1 Sampling

In this chapter, we study the representation of a continuous-time signal by its samples. This is provided in terms of the sampling theorem.

Consider three continuous-time signals $x_1(t), x_2(t)$, and $x_3(t)$shown in Figure 2.1. We can choose a sampling rate of T sec. such that $x_1(kT) = x_2(kT) = x_3(kT)$, for $k = 0, \pm 1, \pm 2, \ldots$

In general, there are an infinite number of signals that can generate a given set of samples. The questions is: *under what conditions, a signal $x(t)$ can be recovered uniquely from its samples*?

We now show that if a signal is bandlimited, then it can be recovered exactly from its samples taken at a sampling rate that is at least twice the highest frequency component in the signal. This is given in terms of the *Shanon Sampling Theorem*. We will show that using ideal sampling (impulse train sampling) and using pulse (practical case) sampling.

2.1.a Impulse Sampling

Consider an input signal $x(t)$ with Fourier transform $X(f) = 0$ for $\mid f \mid > W$Hz. The signal is multiplied by the impulse train to obtain a sampled signal $x_s(t)$ as shown in Figure 2.2 (note: still $x_s(t)$ is a continuous-time signal).

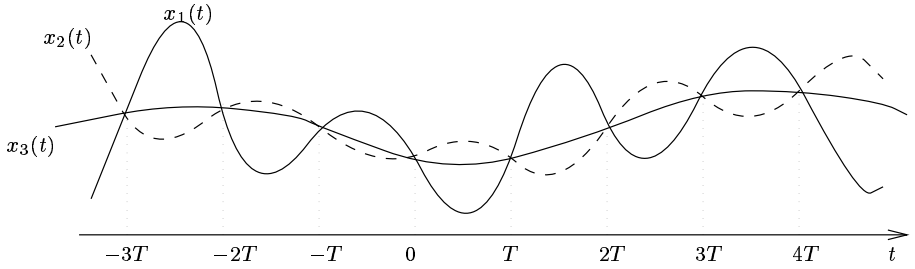

Figure 2.1: Three continuous-time signals

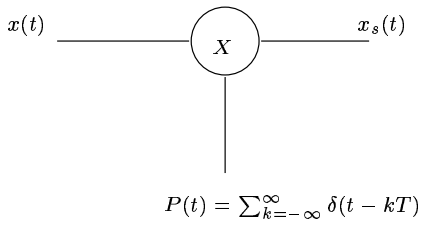

Figure 2.2: Multiply $x(t)$ by the impulse train to obtain a sampled signal $x_s(t)$

$$x_s(t) = x(t)P(t) \tag{2.1}$$

$$X_s(f) = X(f) * P(f) \tag{2.2}$$

$$= \frac{1}{T} \sum_{k=-\infty}^{\infty} X(f - n/T) \tag{2.3}$$

$$X_s(f) \stackrel{\triangle}{=} f_s \sum_{k=-\infty}^{\infty} X(f - nf_s), \tag{2.4}$$

$$\text{where} \quad f_s \stackrel{\triangle}{=} \frac{1}{T} \tag{2.5}$$

That is, $X_s(f)$ is an *a periodic* function of frequency consisting of a summation of shifted replicas of $X(f)$, s called by f_s. We shall denote f_s by the sampling frequency in Hertz(Hz).

To examine the recovery issue we will examine three cases:

$$\begin{array}{ll} f_s > 2W; & \text{Over sampling} \\ f_s = 2W; & \text{Perfect sampling} \\ f_s < 2W; & \text{Under sampling} \end{array}$$

as shown in Figure 2.3

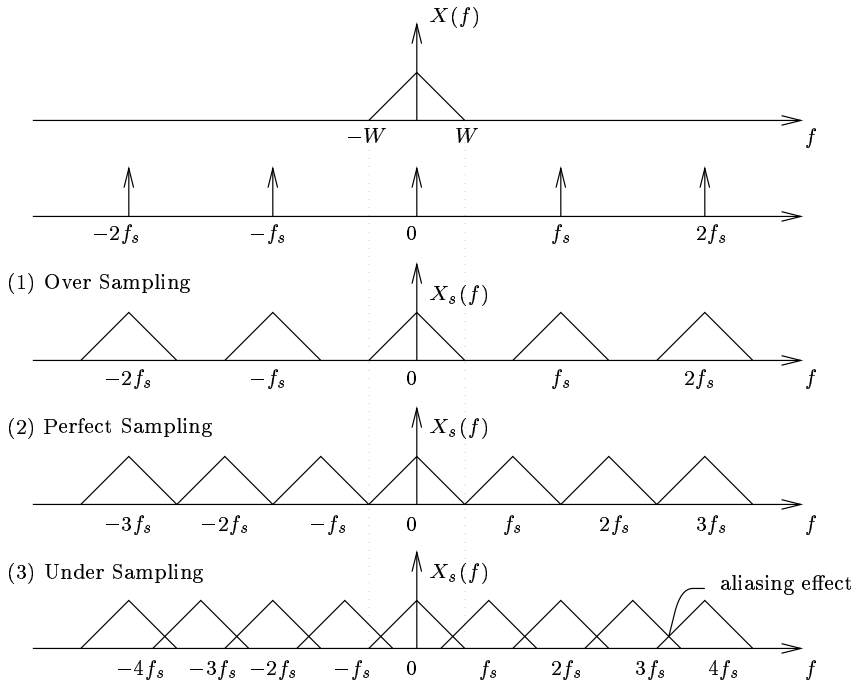

Figure 2.3: Three cases of sampling

For exact reconstruction of a bandlimited signal with highest frequency component at $f = WH_z$, we need to sample the time-signal at a rate $f_s \geq 2WH_z$ as shown in Figure 2.4.

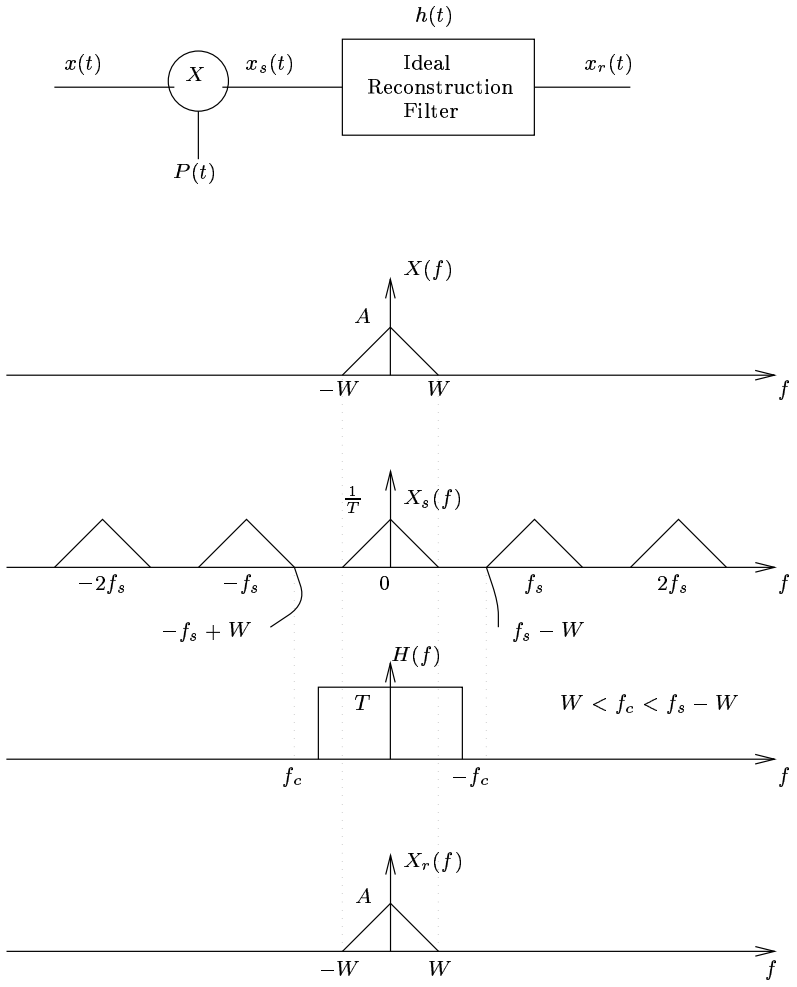

Figure 2.4: Reconstruction

2.1.b Pulse Sampling

$$x_s(t) = p(t)x(t) \tag{2.6}$$

$$p(t) = \begin{cases} 1, & \mid t - nt \mid < \alpha/2T \\ 0, & \text{elsewhere} \end{cases}$$

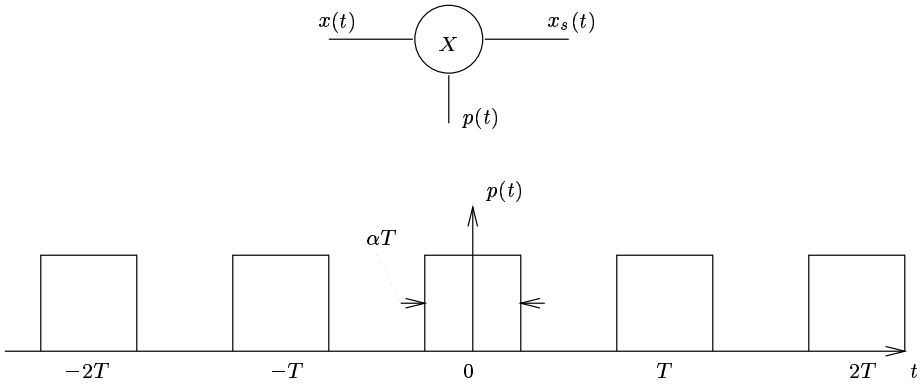

Figure 2.5: Pulse Sampling

The pulse train $p(t)$ is a periodic function, hence

$$p(t) = \sum_{k=-\alpha}^{\infty} C_k e^{j\frac{2\pi}{T}kt} \tag{2.7}$$

$$C_k = \frac{1}{T}\int_{\frac{-\alpha T}{2}}^{\frac{\alpha T}{2}} 1e^{-j\frac{2\pi}{T}kt}dt = -\frac{1}{T}\left\{\frac{1}{j\frac{2\pi}{T}k}\right\}\left(e^{-j\frac{2\pi}{T}k\frac{\alpha}{2}T} - e^{+j\frac{2\pi}{T}k\frac{\alpha}{2}T}\right) \tag{2.8}$$

$$= \frac{\sin(\pi\alpha k)}{\pi k} \tag{2.9}$$

Hence,

$$C_k = \alpha\frac{\sin\pi\alpha k}{\pi\alpha k} = \alpha sinc(\alpha k) \tag{2.10}$$

$$p(t) = \sum_{k=-\infty}^{\infty} \alpha sinc(\alpha k)e^{j\frac{2\pi}{T}kt} \tag{2.11}$$

$$x_s(t) = \left(\sum_{k=-\infty}^{\infty} C_k e^{j\frac{2\pi}{T}kt}\right)x(t) \tag{2.12}$$

$$= \sum_{k=-\infty}^{\infty} C_k x(t)e^{j\frac{2\pi k}{T}t}$$

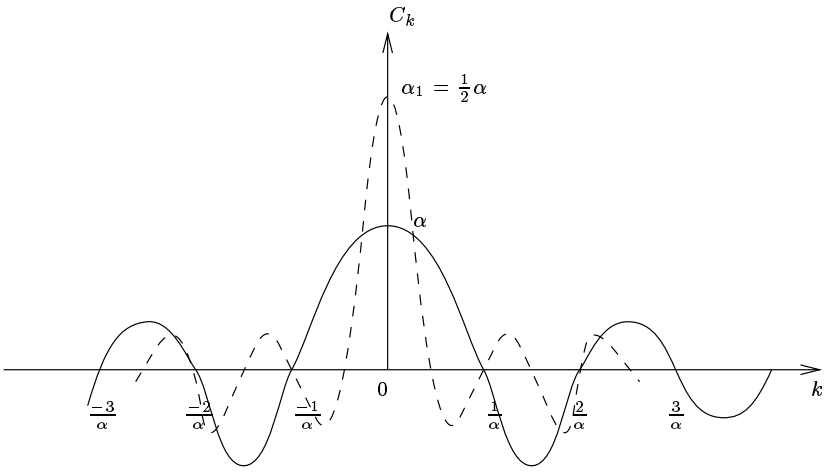

Figure 2.6: Effect of α on C_k

But,

$$x(t) \longleftrightarrow X(f) \tag{2.13}$$
$$x(t-t_0) \longleftrightarrow e^{-j2\pi f t_0} X(f) \tag{2.14}$$
$$e^{\pm j2\pi f_0 t} x(t) \longleftrightarrow X(f \mp f_0) \tag{2.15}$$

Hence,

$$X_s(f) = \sum_{k=-\infty}^{\infty} C_k X(f - \frac{k}{T}) \tag{2.16}$$
$$X_s(f) = \sum_{k=-\infty}^{\infty} C_k X(f - kf_s), \tag{2.17}$$

Where

$$f_s = \frac{1}{T} \tag{2.18}$$

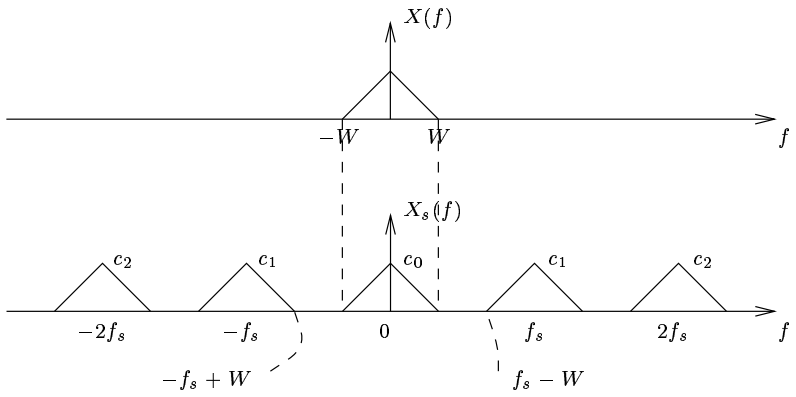

Figure 2.7: Sampling with $f_s < 2W$

Again, we have three cases:

$f_s > 2W$;	Over sampling
$f_s = 2W$;	Perfect sampling
$f_s < 2W$;	Under sampling

For exact reconstruction, we must, again, have $f_s \leq 2W$ and use an ideal lowpass filter with bandwidth such that: $f_c > W$ and $(f_s - W) > f_c$. This is stated in terms of the sampling theorem.

The Sampling Theorem
Let $x(t)$ be a bandlimited signal with $X(f) = 0$ for $\mid f \mid > WH_z$. Then, $x(t)$ is uniquely determined by its samples $x(nT)$, $n = 0, \pm 1, \pm 2, \ldots$ if

$$f_s > 2W \text{ , where } f_s = \tfrac{1}{T},$$

by passing the sampled function through a lowpass filter with cutoff frequency f_c such that: $W < f_c < f_s - W$, and with a Gain of T.

◇ ◇ ◇

The handout on signal reconstruction provides details of the process.

Now, we want to put the samples inside the computer memory. To do that, we have to quantize the samples according to the available accuracy. Therefore, digitizing a signal (image) involves two processes: *sampling* and *quantization*. Quantization is done by assigning a value to each sample according the wordlength n used; for example, if you have n-bits wordlength, we can assign $q = 2^n$ quantization levels. The value q is taken to be equal to the range of $x(t)$; i.e.,

$$q = \mid x_{max}(t) - x_{min}(t) \mid \tag{2.19}$$

Once the signal is quantized (after being sampled), all what we have in the computer is a set of numbers. On these numbers, we are going to build the theory of discrete-time signals and system.

2.2 Comments on Signal Reconstruction

In the following[1], we study two important forms of data reconstruction: interpolation which is suited for data transmission systems and extrapolation which is more suited for feedback systems.

2.2.a Interpolation

Interpolation is the process of fitting a function of time through the data points $x[n] = x(nT)$ of a data sequence $\{x[n]\}$. Any such function $y_i(t)$ is called an interpolation function of the data sequence. It is generally not convenient to fit a polynomial through the data points because the degree of the polynomial must be one less than the number of data points, which is usually very large. Hence, we take a different approach.

Let the interpolation function be given by

$$y_i(t) = \sum_{n=-\infty}^{\infty} x[n] h_i(t - nT) \tag{2.20}$$

This interpolation function is linear in $x[n]$ and will pass through the data points $x[n] = x(nT)$ if

$$h_i(0) = 1 \tag{2.21}$$

$$h_i(kT) = 0, \qquad k \neq 0 \text{ is an integer} \tag{2.22}$$

[1]Adopted from: Elmer G.Gilbert, Notes on sampled-data systems, Computer, Information and Control Program, The University of Michigan, Ann Arbor, 1983.

The form of the interpolation function $y(t)$ depends on the form of the interpolation-generating function $h_i(t)$ in (2.20). Figure 2.8 shows three examples of the function $h_i(t)$ (and $y_i(t)$) out of many other possibilities. The first two generating functions are the first members of a class of generating functions such that:

$$\frac{d^N h_i}{dt^N} = \text{ piecewise constant function,} \tag{2.23}$$

$$h_i(t) = 0, \mid t \mid \geq NT, N \neq 0. \tag{2.24}$$

For $N = 0$, zero-order interpolation, $y_i(t)$ is piecewise constant and for $N = 1$, first-order interpolation, $y_i(t)$ is continuous and piecewise linear. For the second-order interpolation ($N = 2$) $y_i(t)$ is continuous and piecewise quadratic, etc. Thus, by introducing more complex functions $h_i(t)$, smoother interpolation functions are obtained. The interpolation function generated by $sinc(t/T)$ (sometimes called the cardinal interpolation function)

$$h_i(t) = \frac{\sin(\pi t/T)}{\pi t/T} = sinc(t/T) \tag{2.25}$$

is the smoothest of all interpolations in the sense that $y_i(t)$ is bandlimited with bandwidth $\frac{1}{2}f_s$, where $f_s = \frac{1}{T}$ is the sampling frequency in Hertz. The first two interpolation functions in Figure 2.8 are not bandlimited. With the cardinal interpolation function, the function $y_i(t)$ has the property $y_i(t) = x(t)$ if $x(t)$ is bandlimited with bandwidth $< f_s/2$. The other interpolations reconstruct $x(t)$ exactly only if $x(t)$ has a specialized functional form. For example, in the case of the first-order interpolation, $y_i(t) = x(t)$ only if $x(t)$ is a piecewise continuous linear function of t, with all slope discontinuities at $t = nT$.

From practical point of view, the interpolation function described above cannot be realized in the on-line (real time) reconstruction of data sequences. To see the nature of this difficulty, consider the first-order interpolation function (Figure 2.8.b). Before one can begin the interpolation in the interval $nT \leq t \leq (n+1)T$, one must know $x[n]\text{and}x[n+1]$. Thus $x[n+1]$ must be known one full sample period *before* it becomes available. It is possible to achieve a first-order on-line interpolation if a delay of T seconds is introduced, that is, if $y_i(t-T)$ is generated. The delay required for zero-order interpolation is clearly $\frac{T}{2}$; for the higher order it is NT. Thus, the smoother the interpolation becomes, the larger the required delay for on-line realizability. The interpolation produced by the cardinal generating function may not begin until all data points are available since the tails of $h_i(t)$ extends to $t = -\infty$.

For data transmission links where the data is inspected some time after it is received, the interpolation delay is not a serious limitation. If, however, the interpolation process occurs inside a feedback loop, the delay may seriously affect the stability of the feedback system.

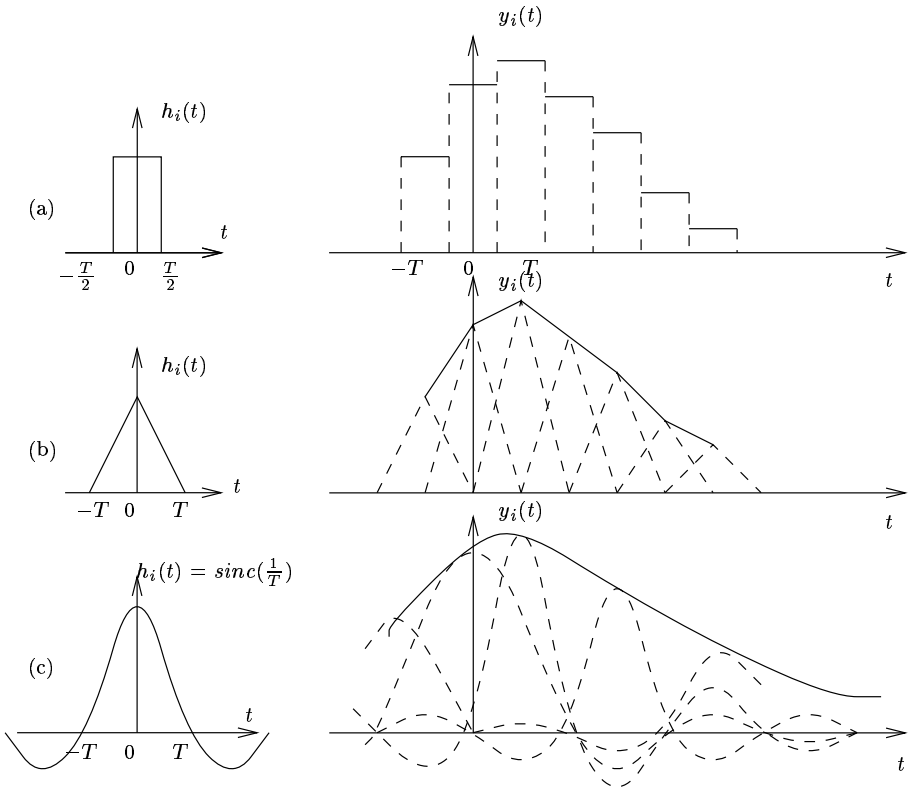

Figure 2.8: Generating Functions and their Resulting Interpolations : (a) Zero-order Interpolation, (b) First-order Interpolation, (c) Cardinal Interpolation

Thus, in a feedback system a different form of data reconstruction called extrapolation is generally employed.

2.2.b Extrapolation

Extrapolation is the process of fitting a function of time to data points received in the past, so that this data sequence may represent the data sequence until the next data point is received. Usually, the extrapolation function $y_e(t)$ is taken to be segments of polynomials in t. Zero-order and first-order extrapolations are shown in Figure 2.9. The generalizations to higher order extrapolations is straightforward, being based on the *Gregory-Newton* extrapolation

formula. It is interesting to note that $y_e(t)$ has jump discontinuities even for higher-order extrapolations. However, for "slowly varying" data sequences the magnitude of the jump generally becomes less as the order of the extrapolation increases. Of course, an n^{th} order extrapolator will reconstruct $x(t)$ exactly if $y_e(t)$ is an n^{th} degree polynomial in t.

A formula of the form (2.20) also holds for extrapolation, that is,

$$y_e(t) = \sum_{n=-\infty}^{\infty} x[n] h_e(t - nT) \tag{2.26}$$

The nature of the extrapolation generating functions $h_e(t)$ is quite different from the interpolation generating functions (see Figure 2.9):

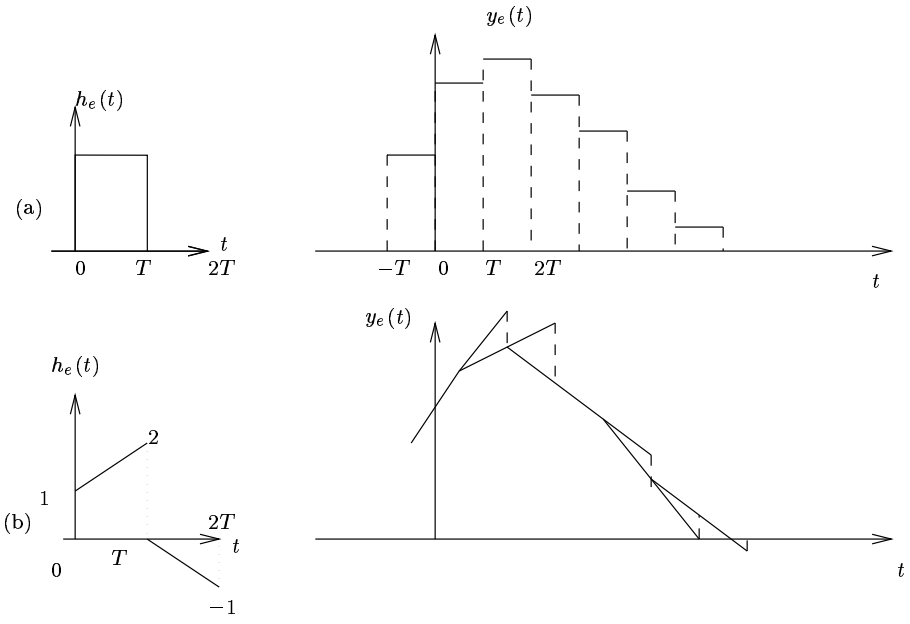

Figure 2.9: Extrapolation Generating Functions and Their Resulting Extrapolations: (a) Zero-Order Hold (b) First-Order Hold

1. Zero-order extrapolator:

$$h_e(t) = \begin{cases} 1 & 0 \le t < T \\ 0 & t < 0, t \ge T. \end{cases} \tag{2.27}$$

2. First-order extrapolator:

$$h_e(t) = \begin{cases} (1+t/T) & 0 \le t < T \\ (1-t/T) & T \le t < 2T \\ 0 & t < 0, t \ge 2T. \end{cases} \tag{2.28}$$

The reader should show that (2.28) is indeed correct.

Since extrapolation begins anew each time a new data point is received, it seems reasonable that the delay introduced by this data reconstruction process should not increase with the order of the extrapolation. Perhaps the best way of seeing this, and also of comparing the processes of interpolation and extrapolation, is to give (2.20) and (2.26) a frequency domain interpretation.

This is done in Figure 2.10. If the function $x(t)$ is passed through an impulse sampler and then through a linear system with impulse response $h_i(t)$, the response of the linear system is $y_i(t)$. The spectrum of $y_i(t)$ is obtained from $X_s(f)$ by

$$Y_i(f) = X_s(f)\ H_i(f), \tag{2.29}$$

where $H_i(f)$ is the transfer function corresponding to the impulse response $h_i(t)$ and is given by

$$H_i(f) = \int_{-\infty}^{\infty} h_i(t) e^{-j2\pi ft} dt. \tag{2.30}$$

For extrapolation the results are the same except the subscript i is replaced by e.

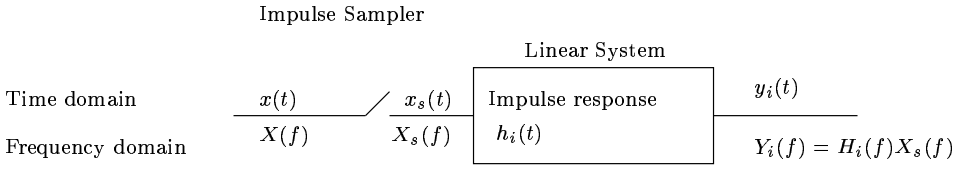

Figure 2.10: Representation of Interpolation as Filtering of $x_s(t)$ where $x_s(t)$ is Obtained by Impulse Sampling

In the case of the cardinal interpolation function we can easily show that

$$h_i(f) = \begin{cases} T & |\ f\ | \le f_s/2 \\ 0 & |\ f\ | > f_s/2. \end{cases} \tag{2.31}$$

Thus, the linear system is an ideal low-pass filter and the interpolation function is an ideal low-pass filter. It is, therefore, apparent why the cardinal interpolation is an exact reconstruction of $x(t)$ when $x(t)$ is bandlimited with bandwidth less than $\frac{1}{2}f_s$.

For the zero-order interpolator it is easy to show that:

$$H_i(f) = T\frac{sin(\pi fT)}{\pi fT} = Tsinc(fT). \tag{2.32}$$

Thus, for this type of interpolation the situation shown in Figure 2.11 results.

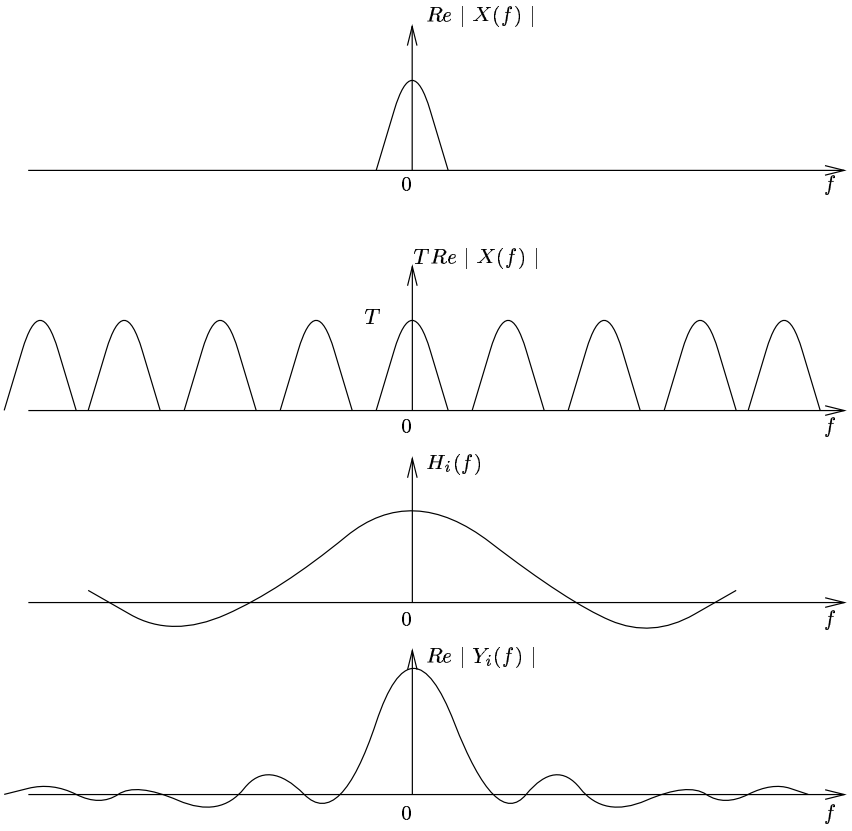

Figure 2.11: Frequency Domain Interpretation of the Interpolation Process (Zero-Order)

Figure 2.12. shows the gain and phase shift associated with interpolations and extrapolations of the zero- and first-order type. In order to make the interpolation realizable of

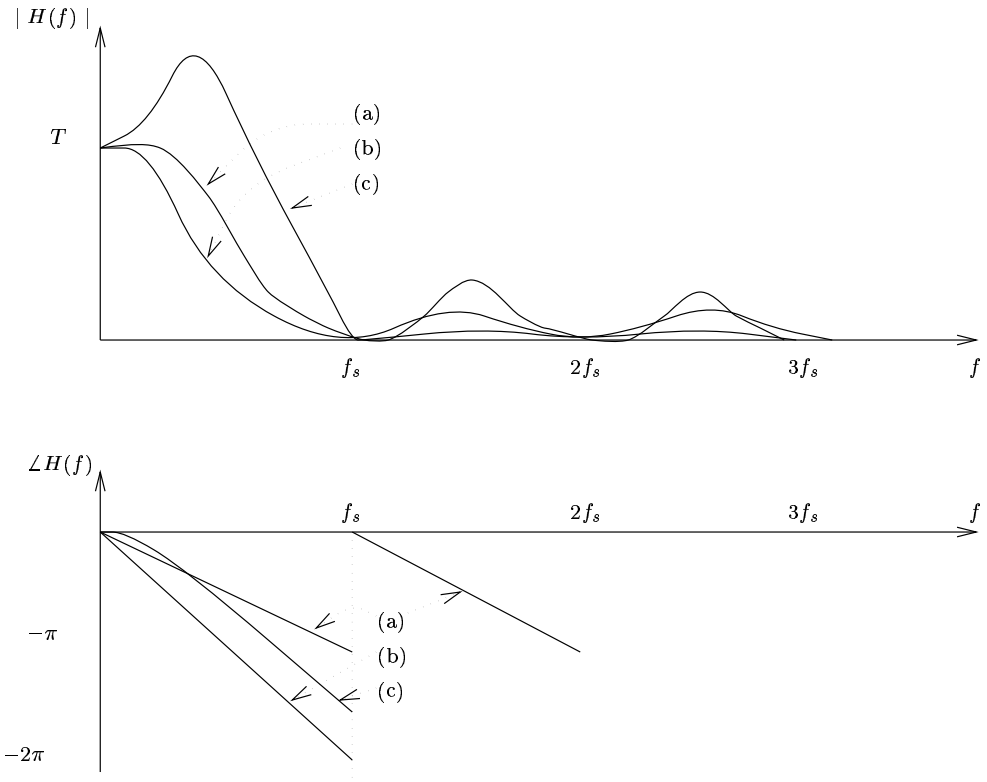

Figure 2.12: Gain and Phase Shift for Zero-Order and First-Order Interpolations and Extrapolators

on-line reconstruction of sampled-data, appropriate delays have been introduced. In each case it can be seen that $H(f)$ has a lowpass character which tends to reject the spectral components of $X_s(f)$ at frequencies greater than $\frac{1}{2}f_s$ which are introduced by sampling. The delayed interpolation of zero-order is exactly the same as the zero-order extrapolation. The Fourier transform of the zero- and first-order interpolator/extrapolator are

(a) Zero-order interpolator and extrapolator: $h_{ir}(t) = h_{er}(t) = h_i(t - T/2)$

$$H_{ir}(f) = H_{er}(f) = H_i(f)e^{-j2\pi fT/2} = T\ sinc(fT)e^{-j2\pi fT/2} \tag{2.33}$$

(b) First-order interpolator: $h_{ir}(t) = h_i(t - T)$

$$H_{ir}(f) = H_i(f)e^{-j2\pi fT} = Tsinc^2(fT)e^{-j2\pi fT} \tag{2.34}$$

(c) First-order extrapolator: $h_{er}(t) = h_e(t - T)$

$$H_{er}(f) = T(1 + j2\pi fT)\ sinc^2(fT)e^{-j2\pi fT} \tag{2.35}$$

We added the extra script "r" in 2.33 and 2.34 to emphasize that these formulas correspond to the realizable interpolator/ extrapolator cases.

As shown in Figure 2.12., if the phase lag of the first-order extrapolation is compared to that of zero-order extrapolation, it is seen to be less for low frequencies but greater for high frequencies. The first-order extrapolation produces less phase lag than the first-order interpolation. Similar results are obtained for higher-order interpolations and extrapolations.

Other types of data reconstruction processes arise in sampled-data systems. Often they may be represented by the general procedures described above. For example, pulse modulation may be approximated by

$$y_p(t) = \sum_{n=-\infty}^{\infty} h_p(t - nT)x(nT), \tag{2.36}$$

where

$$h_p(t) = \begin{cases} 1 & -\alpha T/2 \leq t \leq \alpha T/2 \\ 0 & \mid t \mid > \alpha T/2. \end{cases} \tag{2.37}$$

The approximation is good if $\alpha << 1$ or $x(t)$ varies little in one sample period.

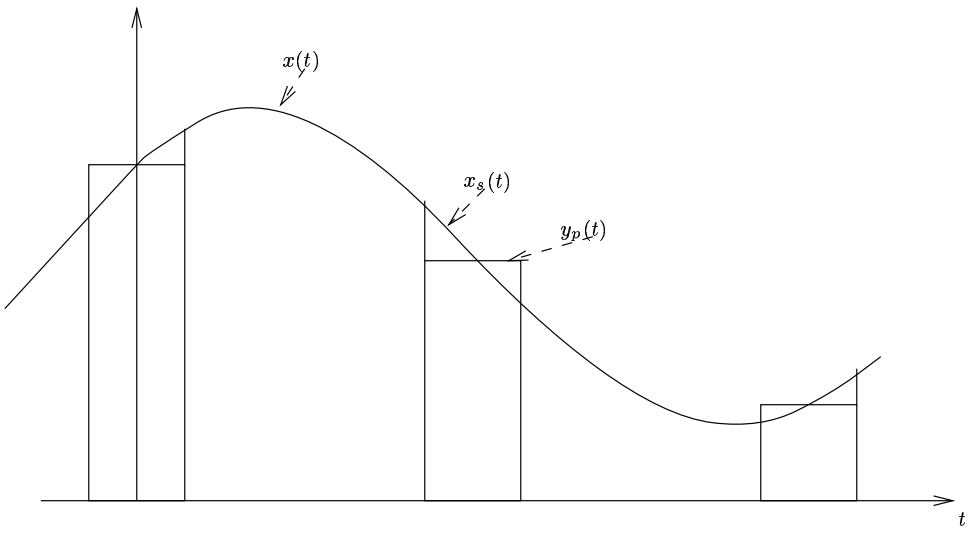

Figure 2.13: Approximation of $x_s(t)$ by $y_p(t)$

See Figure 2.13. Another example is the use of a Butterworth lowpass filter for interpolation. If the lowpass filter is preceded by a zero-order extrapolator (or zero-order interpolator), the resulting $H_i(f)$ has a gain given by

$$\mid H_i(f)\mid = \mid Tsinc(fT)\mid \cdot \left[1+\left(\frac{f}{f_c}\right)^{2k}\right]^{-\frac{1}{2}}, \tag{2.38}$$

where f_c is the cutoff frequency of the filter and k the order of the filter.

2.2.c Implementation of Sampling and Data Reconstruction Processes

The physical devices which implement the processes of sampling and data reconstruction may assume different forms, depending on such things as: the physical nature of the physical signal involved, the required accuracy, cost, weight, reliability, etc. Since a detailed treatment of these matters is not feasible in these notes, let us consider in a limited way the special problems of transferring continuous data, in the form of voltages, into and out of a digital computer.

A scheme for entering data is shown in Figure 2.14. The sample-hold device picks up the samples of the input voltage $x(nT)$ and holds them a short time so that the analog-to-digital conversion may be completed. If conversion of $x(t)$ is made directly, the variability of conversion time with the input voltage would cause the sampling instants to be spaced non-uniformly in a complex way.

The arrangement shown in Figure 2.14 includes a very simple mechanization for the sample-hold device. The switch is closed for $nT - \beta T \leq t < nT$. During this period $y(t) = x(t)$. Thus, when the switch is open, $nT < t < nT + T - \beta T$, $y(t) = x(nT)$. If $\beta << 1$, the sample-hold device has an output which approximates that of a zero-order extrapolator. For low sampling rates the switch might be a computer operated relay; for high sampling rate it might be a transistor circuit. It is important that the analog-to-digital converter (A/D) have a high input impedance so that the capacitor does not discharge appreciably when the switch is open.

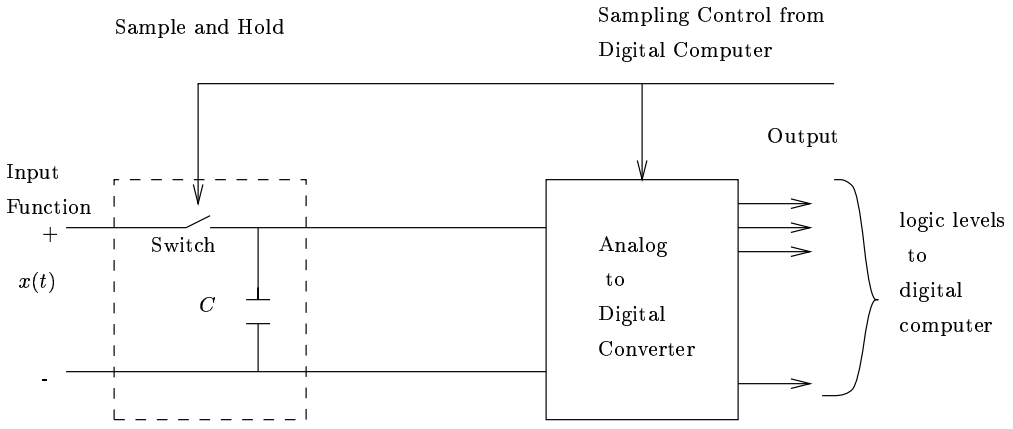

Figure 2.14: Arrangement for Entering Sampled Data into Digital Computer

There are a variety of ways in which the details of the A/D may be mechanized. In principle, however, all converters consist of three basic elements: a voltage source, controlled by digital elements; a comparator, which determines the sign of the difference between the input voltage and the output of the voltage source; and a control device, which operates the digital elements from the comparator output. The control device provides a systematic procedure for adjusting the voltage source so that its output matches the input voltage. When the match is obtained the state of the digital elements is a digital representation of the input voltage, which may be read into the computer. At best, the accuracy of the conversion is determined by the number of bits or decimal digits carried in the representation. This round-off error is called the quantizing error of the conversion.

The effect of quantizing is shown graphically in Figure 2.15. Clearly, the input-output relation is nonlinear. Thus, any system which utilizes conversion equipment is nonlinear and the theory of sampled-data systems is not applicable! Often, however, the quantizing errors are so small that the stair-step relation in Figure 2.15. may be approximated by a linear one. The validity of such approximations should always be checked, either by further analysis or by system simulation. One analytic approach, which has met with considerable success, is to approximate the quantizing error as a disturbing random noise. This analysis is out of the scope of this course and will not be presented in these notes.

Let us now consider the process of data reconstruction. The transferring of data out of the digital computer involves digital-to-analog conversion (D/A). A schematic representation of D/A is shown in Figure 2.16. Realization (a) employs a ladder network, while realization (b) utilizes an operational amplifier. The binary representation of $x[n]$ is loaded into the register of the converter at $t = nT$, and is held there until $t = nT + T$ when the next data point is entered. Thus, neglecting the switching transients and quantizing errors, the converter is a direct implementation of a zero-order extrapolator (sometimes called a zero-order hold).

By combining several converters, extrapolators of higher order may be realized. One realization of a first-order extrapolator (first-order hold) is shown in Figure 2.17. At $t = nT$ the switch S is closed for a very brief instant, discharging the capacitor C. Furthermore, the first converter is loaded with $x[n]$, and the second with $-x[n-1]$ (obtained from the register of the first converter). For $T > t - nT > 0$ the output amplifier produces $x[n] + (x[n] + x[n-1])\,(t - nT)/T$ which is the desired extrapolation.

Provided the necessary additional delay for realizability is added, interpolators can be constructed in a similar way (the reader should obtain a realization of the first-order interpolator). In practice, extrapolators and interpolators of order higher than first are rarely employed. Extrapolators of any order may be built utilizing only one D/A converter. For details the reader should consult the immense literature on this subject.

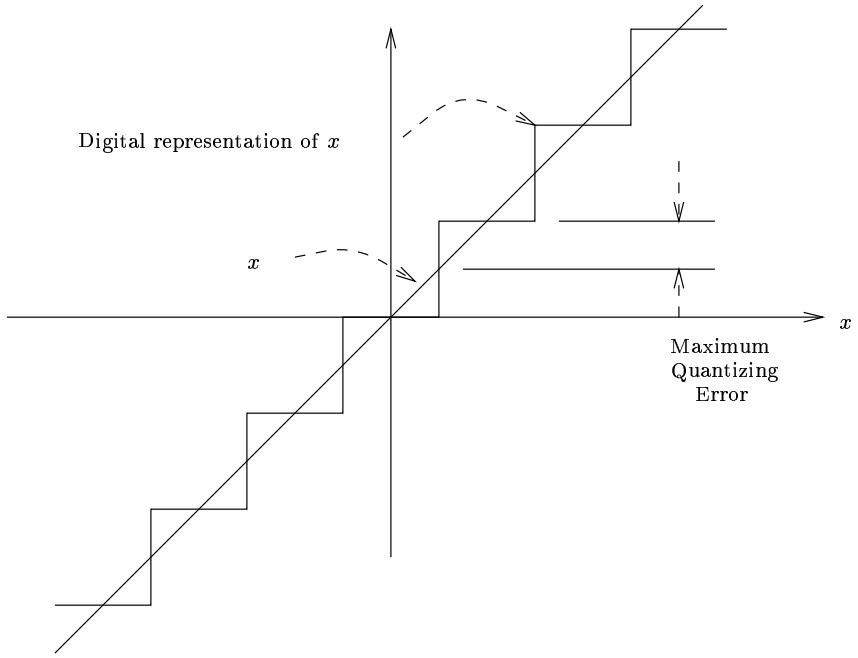

Figure 2.15: Input-Output Relation for Analog-To-Digital Converter (A/D)

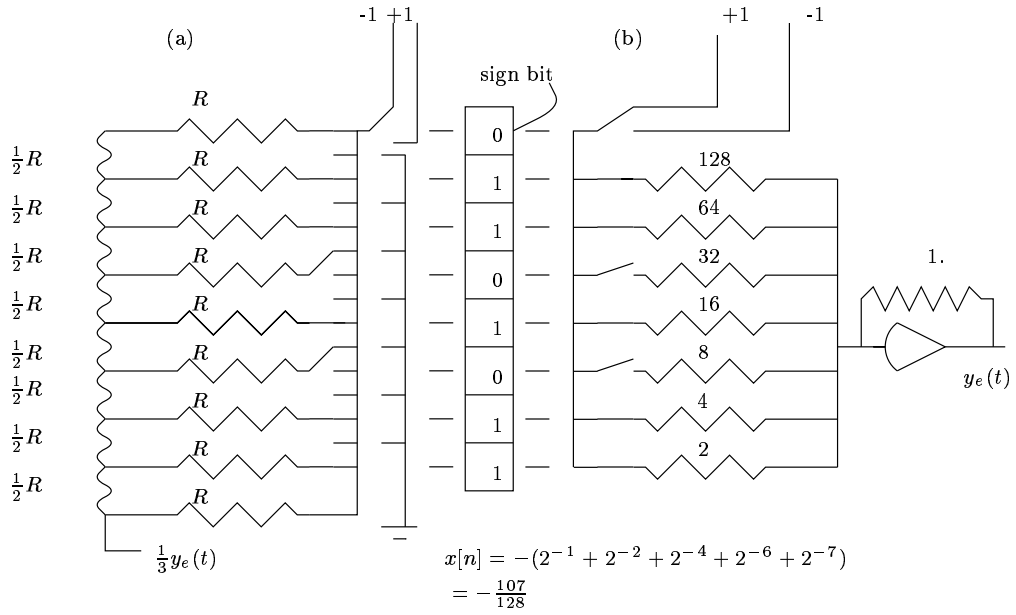

Figure 2.16: Analog-To-Digital Converter (A/D) with Register, (a) Ladder Network, (b) Operational Amplifier

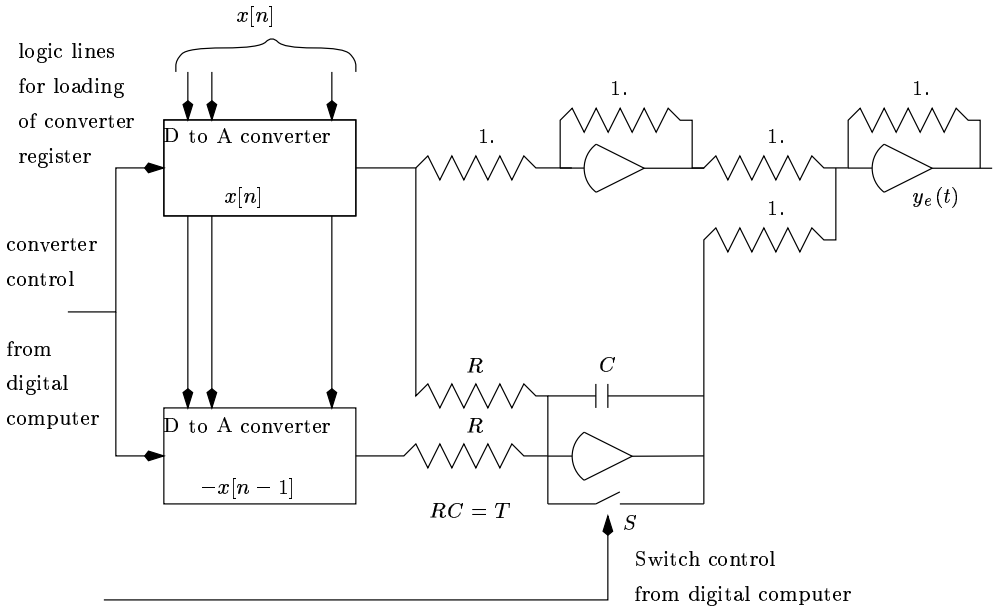

Figure 2.17: Implementation of First-Order Extrapolation