The Discrete Fourier Transform

Part 2

Prof L Gelman

Module "SIGNAL ANALYSIS AND PROCESSING"
NME3523

Analogue Antialiasing Filters

- **Antialiasing filter can be used to avoid aliasing in signals containing many frequencies; this filter is the lowpass (i.e. allows only low frequencies through) filter**

- **An antialiasing filter effectively cuts off frequencies higher than the maximum frequency of interest**

- **This antialiasing filter must be employed *before* the signal is digitized**

- **It is not good trying to use a lowpass filter on the *digitized* signal because the aliasing effects occur *because of the sampling process***

Analogue Antialiasing Filters

- **Any aliasing effects would already be stored in the digitized signal and cannot be removed by low pass filtering**

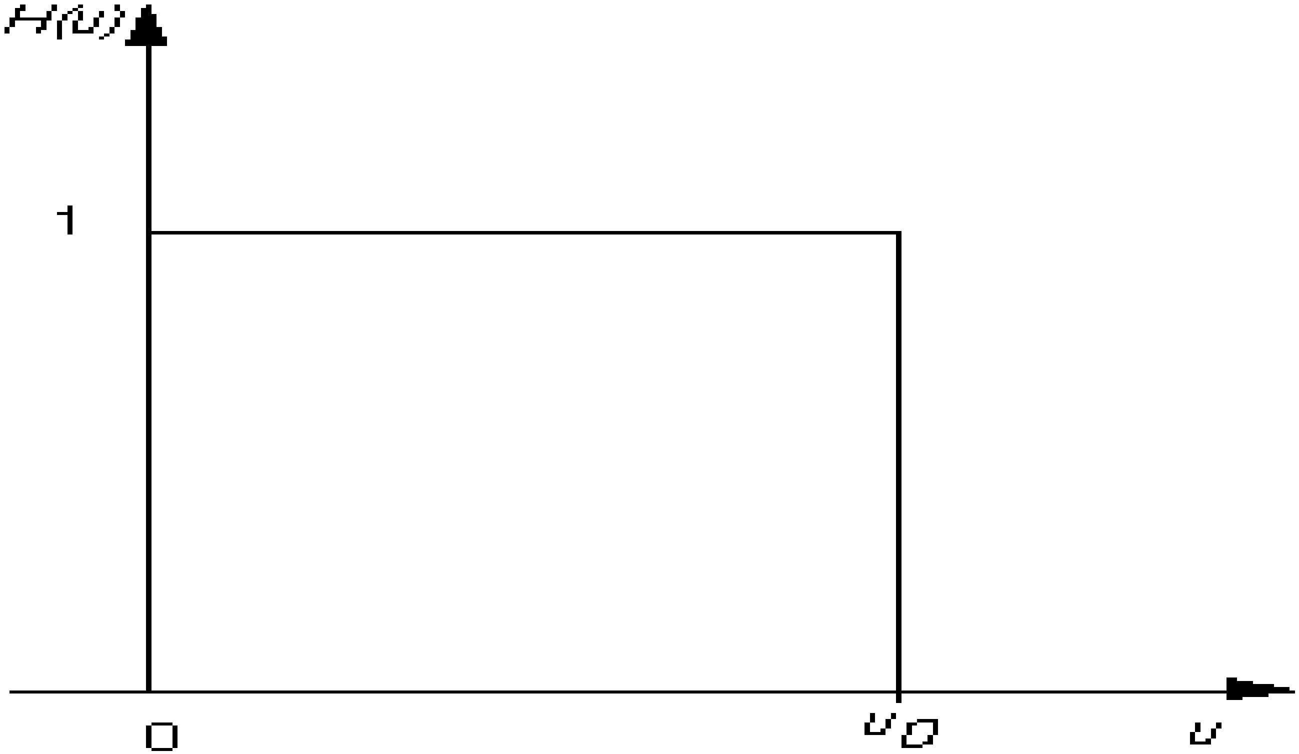
- **Ideally, the anti-aliasing filter should have a lowpass frequency response given by**

$$H(f) = \begin{cases} 1, & |f| \le f_s/2 \\ 0, & |f| \ge f_s/2 \end{cases}$$

- **Such a "brickwall" type frequency response cannot be realized using practical analog circuit components**

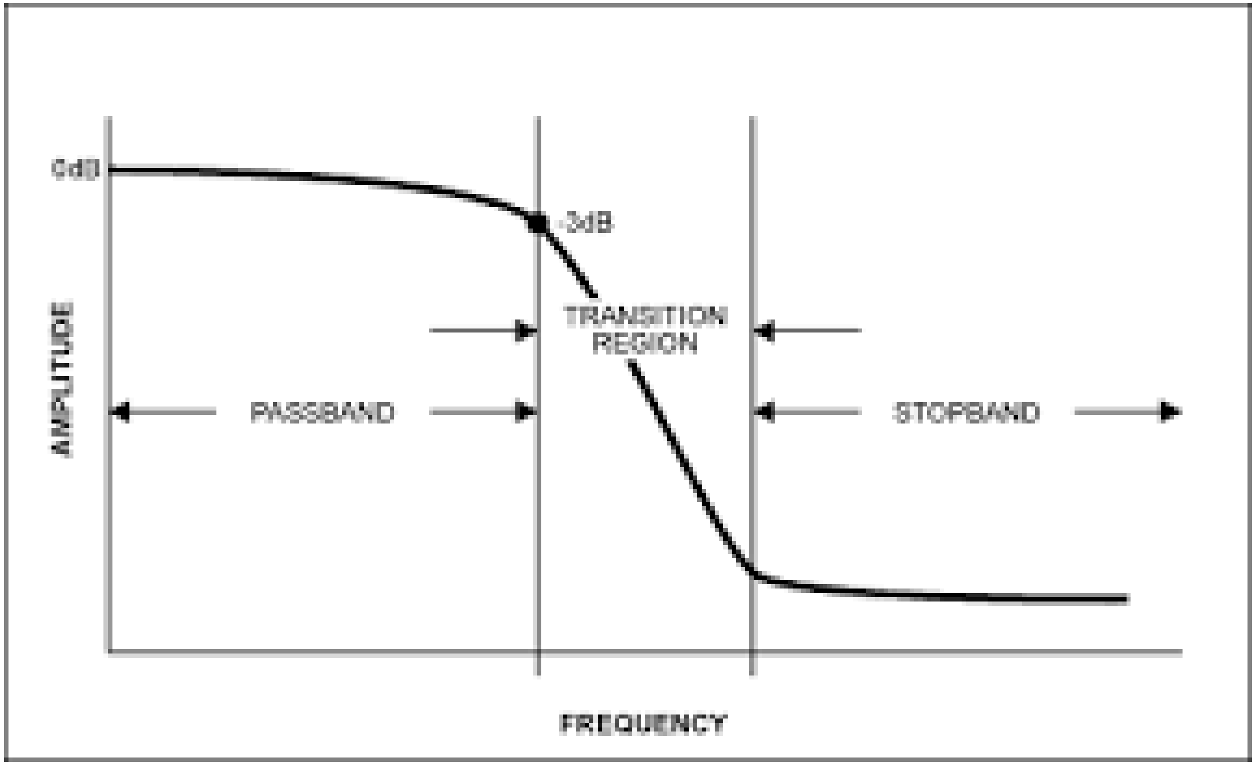
The Ideal Antialiasing Filter

The ideal antialiasing filter is shown below



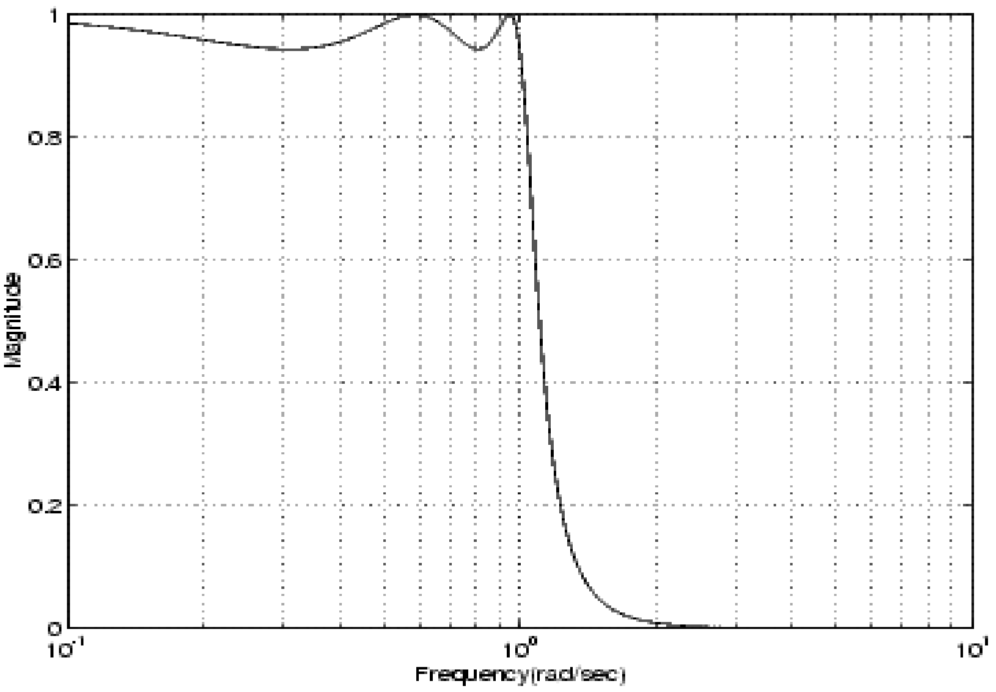
The Real Antialiasing Filter

The real antialiasing filter is shown below

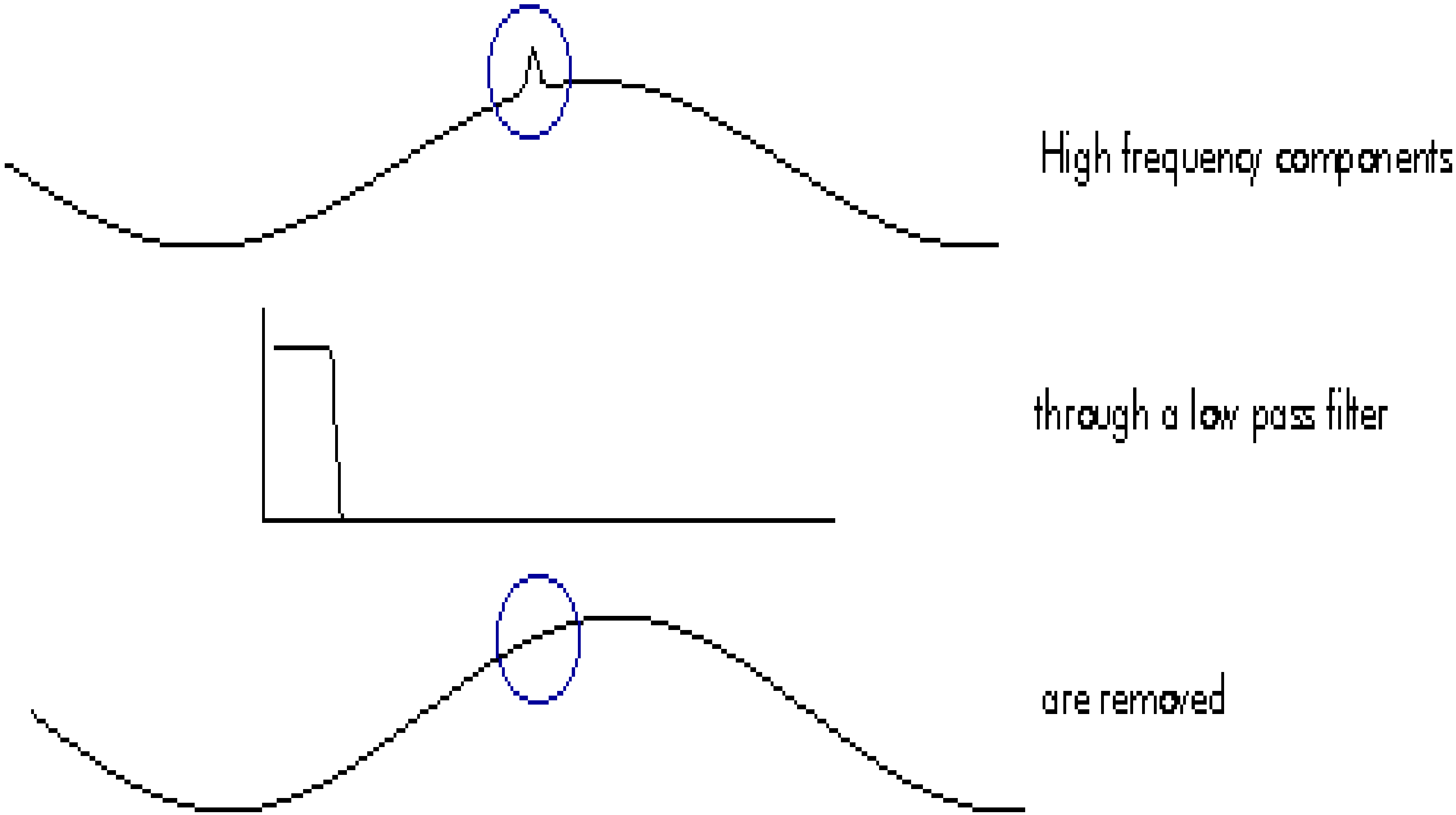


The Chebyshev Analogue Antialiasing Filter

The typical magnitude response for the Chebyshev filter



Analogue Antialiasing Filters



Analogue Antialiasing Filters

- **A practical anti-aliasing filter should have**

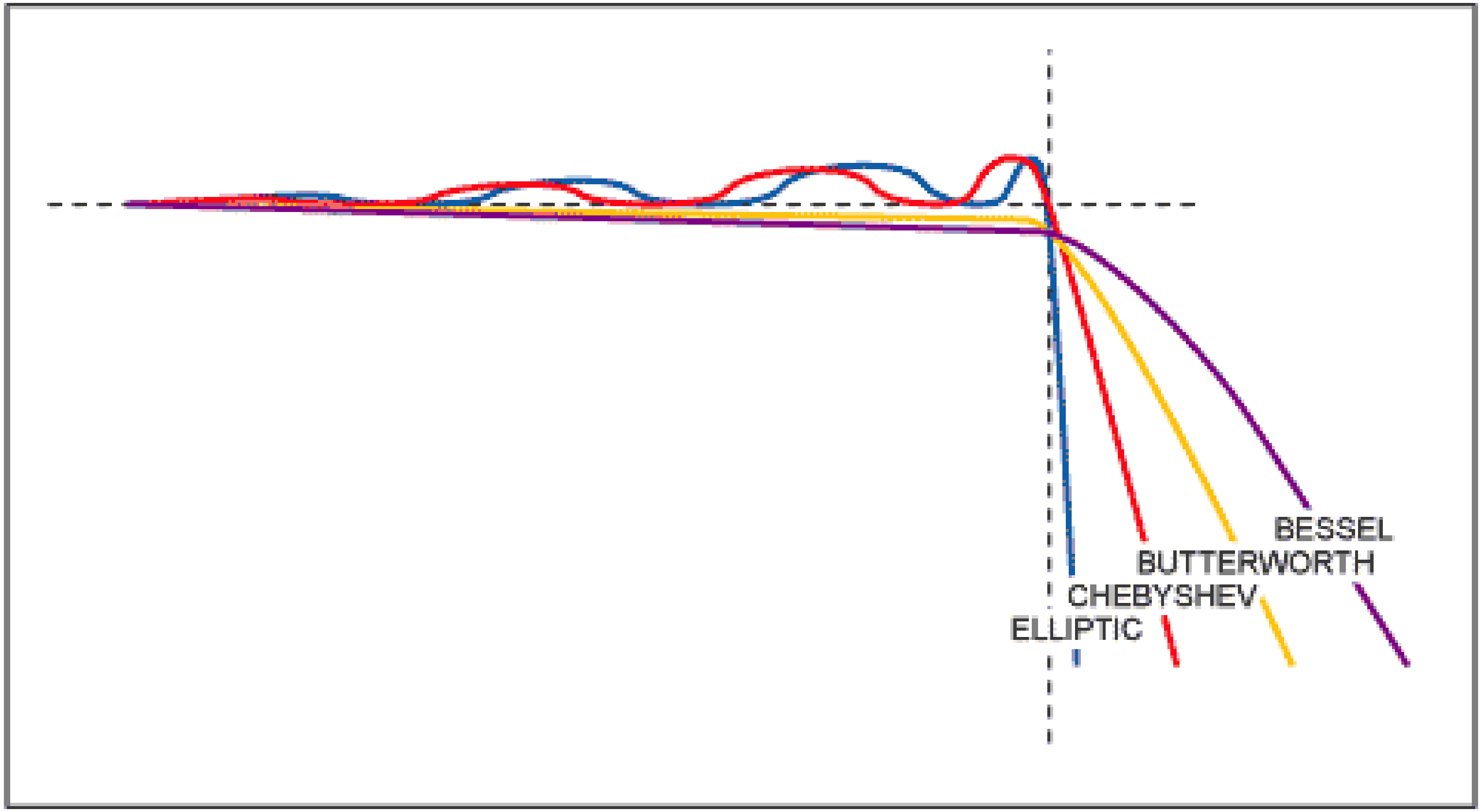
- **a magnitude response approximating unity in the passband with an acceptable tolerance**
- **stopband magnitude response near 0**
- **an acceptable transition band separating the passband and the stopband**

- **The passband edge frequency *Fp* is determined by the *highest frequency* in the signal that must be faithfully preserved in the sampled version**

- **The attenuation level (dB/oct) at frequencies greater than**

  ***Fp* is determined by the *amount of aliasing that can be tolerated* in the passband**

Analogue Antialiasing Filters

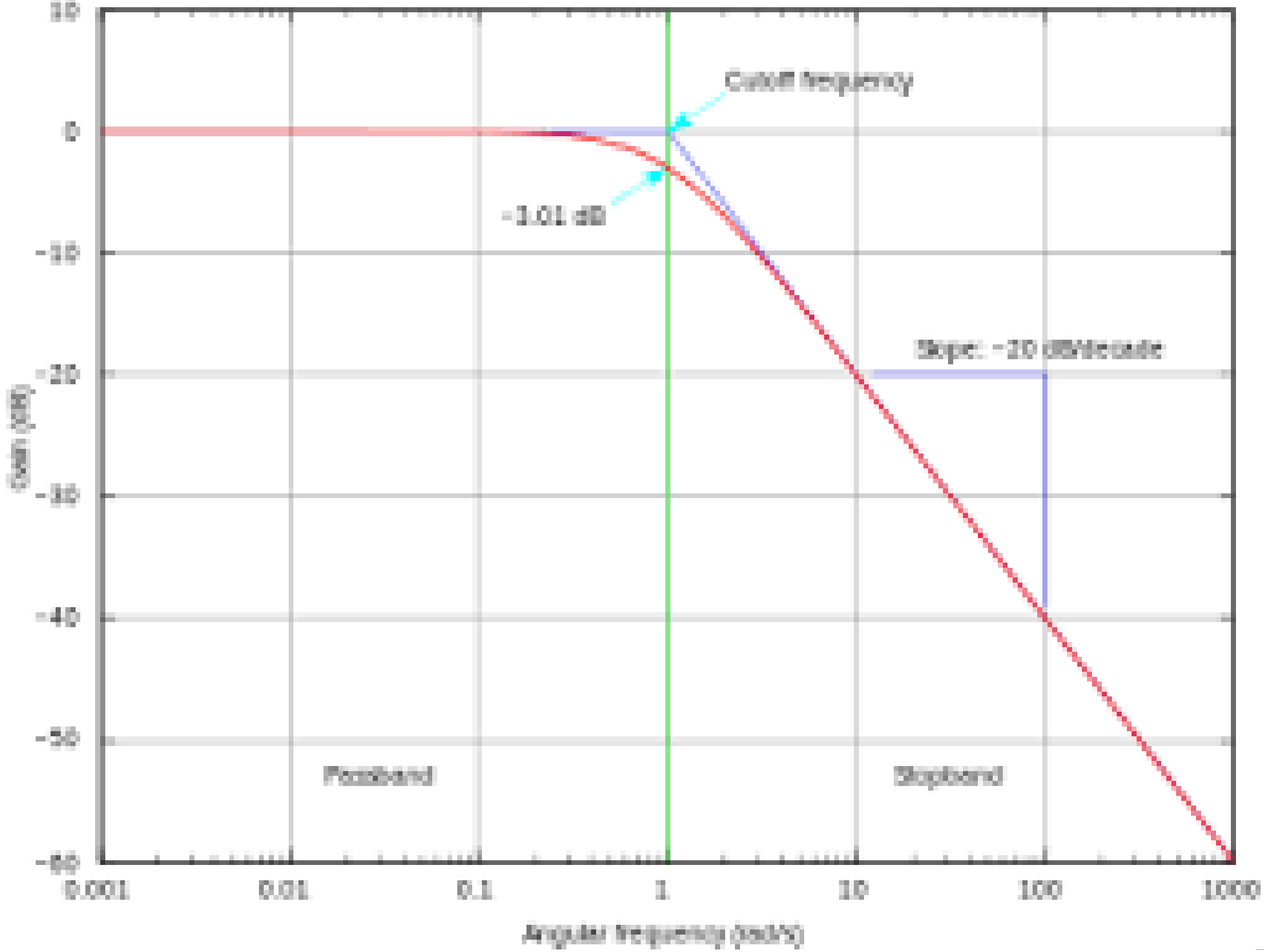


Analogue Antialiasing Filters

- **In applications requiring minimal aliasing, the sampling frequency is typically chosen to be *3 to 4 times* the passband frequency of the analogue antialiasing filter**

- **Selection of the coefficient depends on the filter attenuation level in the transition band**

Analogue Antialiasing Filters



Digital Antialiasing Filters

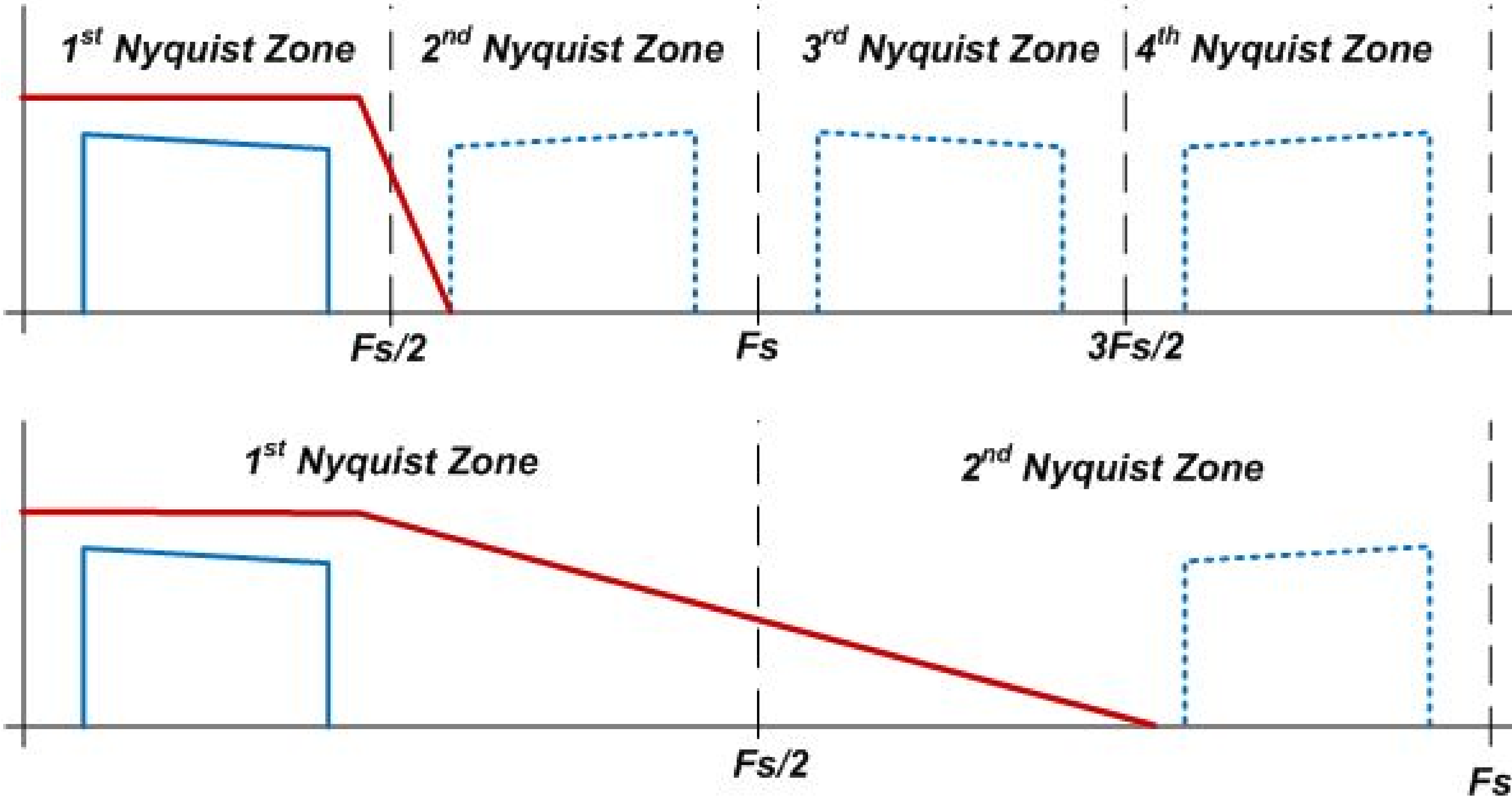
- **One way to reduce the requirements of the analogue anti-aliasing filter is to use *over-sampling***

- **In this case, the signal is sampled with a considerably higher rate (up to ten times) than required to fulfill the Nyquist criteria**

- **Hence, the distance to the first mirrored Fourier transform on the frequency axis will be much longer than if sampling were performed using Nyquist rate**

- **The down-sampling process can be implemented completely in the digital domain by first passing the high sampled digital signal through a digital anti-aliasing filter**

Signal Oversampling



Digital Antialiasing Filters

- **Designing a digital filter having the required passband characteristics is not very hard task**

- **At last, following the digital anti-aliasing filter is a decimator or down-sampler**

- **To perform the downsampling, the downsampler only passes every D-th sample from input to output and ignores the others**

The Fast Fourier Transform

Fast Fourier Transform Algorithms

- **It is known that the direct DFT computation requires** N^2 **complex multiplies and** $N(N-1)$ **complex additions**

- **This computation is inefficient because the symmetry and periodicity properties of the DFT complex factors are not utilised**

- **One of the *most significant discoveries* in the filed of digital signal processing was the fast Fourier transform (FFT), a set of algorithms that can evaluate DFT or IDFT with number of multiplication operations proportional to** $N\log_2 N$ **and to** $(N/2)\log_2 N$ **when** N **is a power of 2, rather than** N^2

- **This represents a *tremendous* decrease in complexity**

Fast Fourier Transform Algorithms

Note that number of additions is also *less*, i.e. $N \log_2 N$ **when** N **is a power of 2**

Fast Fourier Transform Algorithms

Comparison of computational complexity for the direct computation of the DFT vs. the FFT is shown in the table

Sequence length N	**Gain in computation complexity**
64	21.3
256	64
1024	204.8
4096	682.6
16384	2340.6
65536	8192

Fast Fourier Transform Algorithms

- **Moreover there is the added bonus of an *increase in accuracy*. Since *fewer* operations have to be performed by computer, round-off errors due to the truncation of products by the limited word size of the computer are reduced, and accuracy is accordingly increased**

- **The basic approach (named "divide-and-conquer" approach) behind all fast algorithms for computing the DFT is to decompose the N point DFT computation into computation of smaller size DFTs and to take advantage of the periodicity and symmetry of the complex function W_N^{kn}**

- **There are many symmetries and periodicities in the DFT matrix and, therefore, many multiplications could be avoided**

Fast Fourier Transform Algorithms

- **Symmetry and periodicity properties of complex function can be written in the form respectively**

$$W_N^{k+N/2} = -W_N^{k}$$

$$W_N^{k+N} = W_N^{k}$$

- **Today, there is an enormous number of fast algorithms for the computation of the DFT, which are based on the mentioned idea**

MATLAB FFT Functions

- **The following functions are included in the MATLAB package for the computation of DFT: fft (x) and fft (x, N).**
- **The function fft (x) computes the DFT (using FFT) of a vector of the same length as that of x**
- **For computing the DFT of a specific length N, the function fff (x, N) is used**
- **Here, if the length of x is greater than N , it is truncated to the first N samples, whereas if the length of x is less than N, the vector x is zero-padded at the end**

MATLAB FFT Algorithms

- **MATLAB uses two different FFT algorithms for cases when the sequence length of is**

- **a power of 2 (radix-2 algorithm is used)**
- **is not a power of 2 (mixed-radix algorithm is used)**

Radix-2 Algorithm

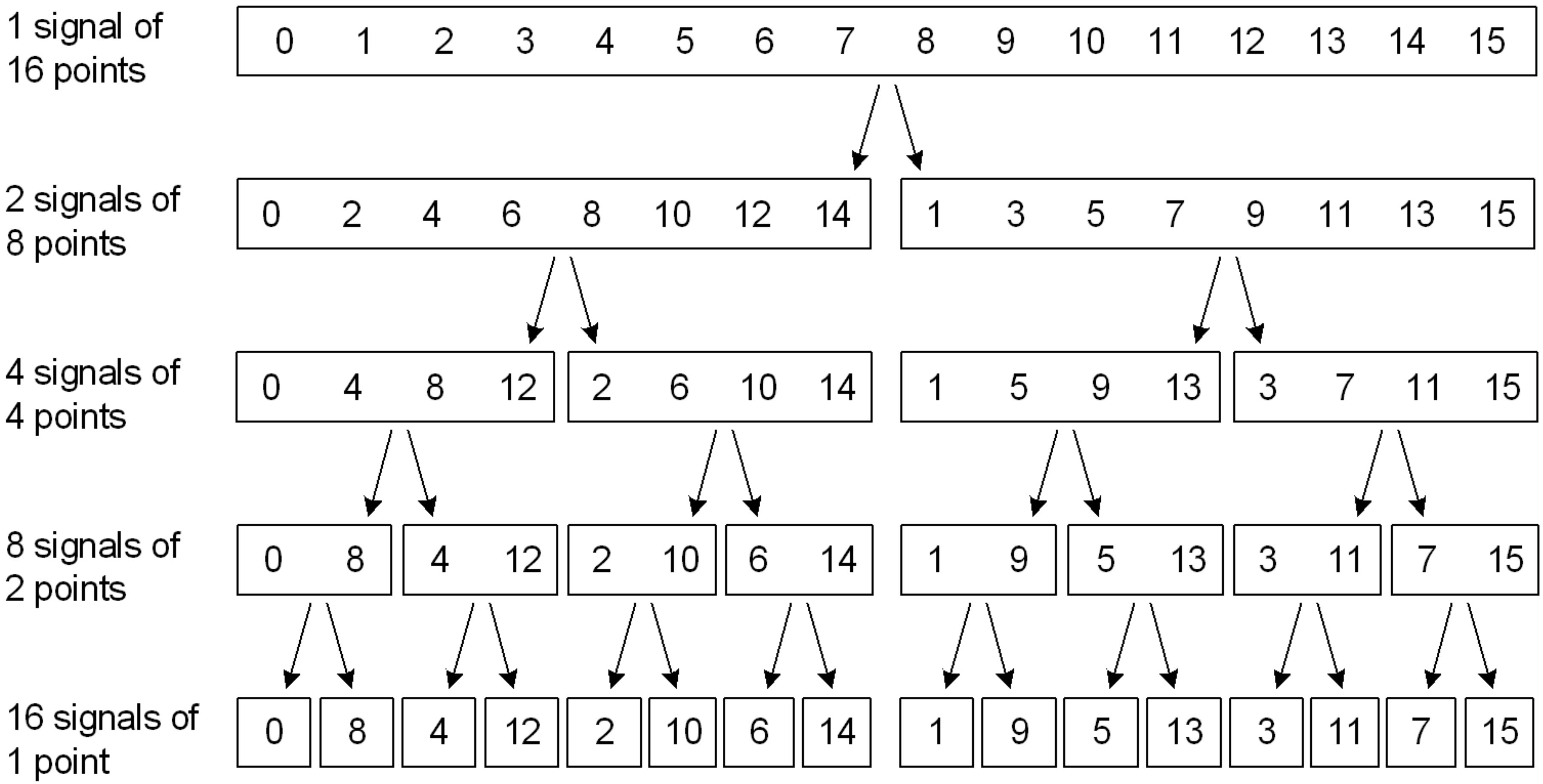
- **The basic realization behind a radix-2 FFT algorithm can be presented as follows**

- **One** N **point DFT can be subdivided into two** $N/2$ **point DFTs**

- **Each** $N/2$ **point DFT can be subdivided into two** $N/4$ **point DFTs and so on until *2 point DFTs are obtained* (2 is the radix of this algorithm)**

- **Only the DFTs of a shorter sequences are worked out; these DFTs are combined together in an ingenious way by FFT algorithm to yield the full DFT**

Radix-2 Algorithm



Radix-4 Algorithm

- **When the number of data points is a power of 4, we can, of course, always use a radix-2 algorithm.**

- **For this case a radix-4 algorithm was proposed, e.g. one N point DFT can be subdivided into four $N/4$ point DFTs, each $N/4$ point DFT can be subdivided into four $N/16$ point DFTs and so on until 4 point DFTs are obtained**

- **It was shown for a radix-4 algorithm in comparison with radix-2 algorithm that**

- **the number of multiplications is reduced by 25%**
- **however, the number of additions is increased by 50%**

Radix-4 Algorithm

We note, that the multiplication reduction is *now less significant* as, with the ready availability of integrated high speed DSP chips, the *total number of operations* is now the more important parameter.

FFT for Arbitrary Signal Length

- **Efficient algorithms for the computation of DFTs having an arbitrary length are possible. In such cases, we can decompose N as a product of factors:**

$$N = N_1 N_2 N_3 \dots N_l = N_1 N_\Sigma$$

- **i. e. initially divide the input sequence into N_1 sequences of length N_Σ**

- **Then we can use appropriate formulae devised to relate the DFTs of the separated sequences to the DFT of the original sequence**

FFT for Arbitrary Signal Length

- **For instance if $N=15$, the original sequence could be separated onto three sub-sequences of five terms each or five sub-sequences of three terms each**

- **We should, as a rule of thumb, divide into sub-sequences as small as possible**

- **In practice, whenever possible, the input sequences should be zero-padded to force N to be *a power of two* sequence**

FFT Algorithms

- **It usually takes a *much longer time* to compute the DFT of a sequence with a non power of 2 lengths than those of a sequence of a power of 2 lengths that is closest to *N***

- **However, the FFT algorithm *also works efficiently if N is not power of 2***

- **Although the total number of operations is very important benchmark, there are other issues to be considered in practical implementation of FFT algorithms**

- **These include the architecture of the processor, the data structure, etc.**

FFT for Special Signal Processors

- **For general-purpose computers, where the cost of the numerical operations dominates, almost all FFT algorithms are good candidates**

- **However, in the case of *special-purpose digital signal processors,* featuring a high degree of parallelism, the structural regularity of the FFT algorithm is *equally important* as arithmetic complexity**

- **Therefore, the irregular structures of some FFT algorithms may render it less suitable for these processors**
- **Structural regularity is also important in the implementation of FFT algorithms on parallel processors**

FFT for the Inverse DFT

- **Although the FFT algorithms discussed previously were presented in the context of computing the DFT, they can also be used to compute the inverse DFT**

- **The only difference between the two transforms is the normalization factor** $1/N$ **and the sign of the phase factor** W_N

FFT Applications

There are four major applications for the FFT processors:

- **the spectral analysis**
- **correlation estimation**
- **fast convolution and the deconvolution**
- **filtering without the convolution and the deconvolution**

Thank you