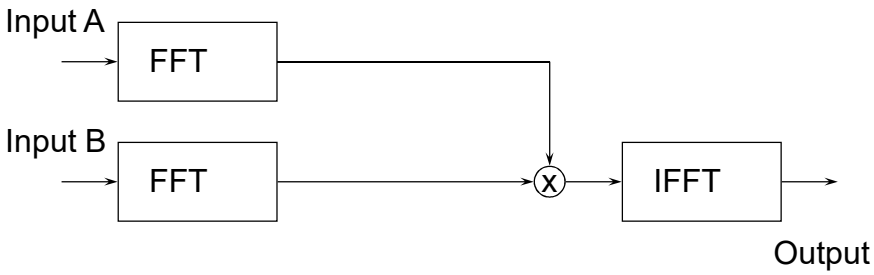
Frequency Domain Convolution And Filtering

jedwards@numerix-dsp.com

Frequency Domain Convolution and Filtering

- Convolution in time domain = multiplication in frequency domain and V.V.
- FFT Length >= N + M - 1
  - Therefore require zero-padding
  - Otherwise circular
- Large computational savings
- Pre-compute convolution kernel FFT for more efficiency

Frequency Domain Convolution

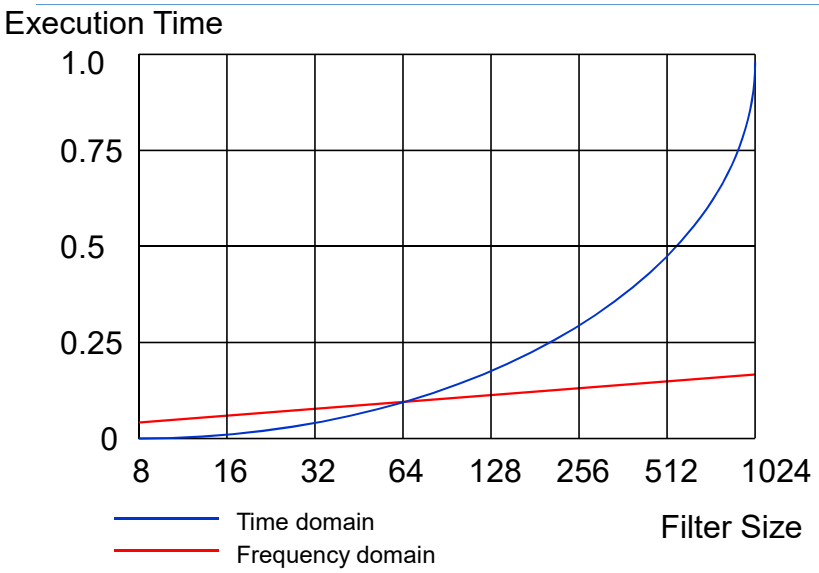


The FFT of a system (e.g. a filter) impulse response is the transfer function.

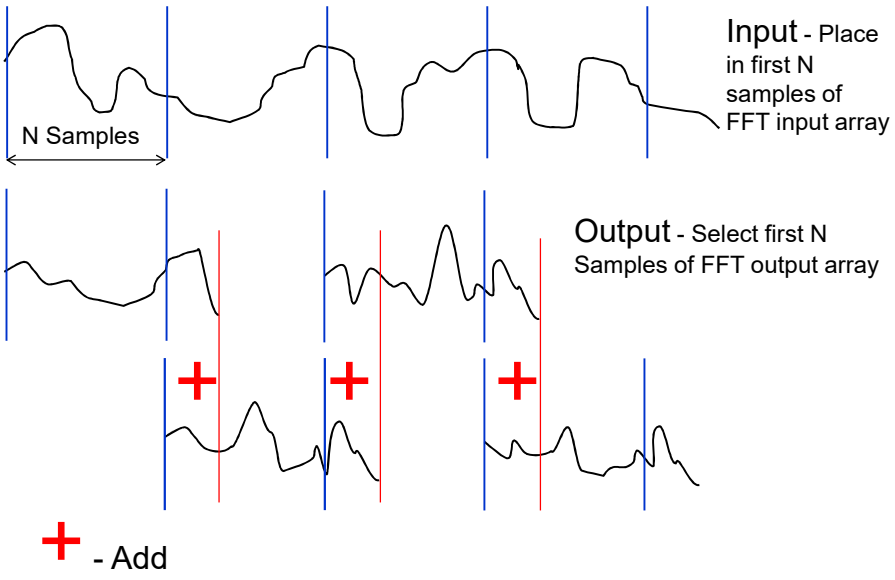
Frequency Domain Filtering

- The finite length transfer function is applied to an “infinite length” signal
- An overlap method will be required
  - Overlap and add
  - Overlap and save
- For a Low-pass filter can not just set high frequency coefficients to zero
  - Generates a time domain $sin(x)/x$ function
- Remove bit reversal
  - Use decimation in frequency FFT
  - Use decimation in time IFFT

Filter Execution Times



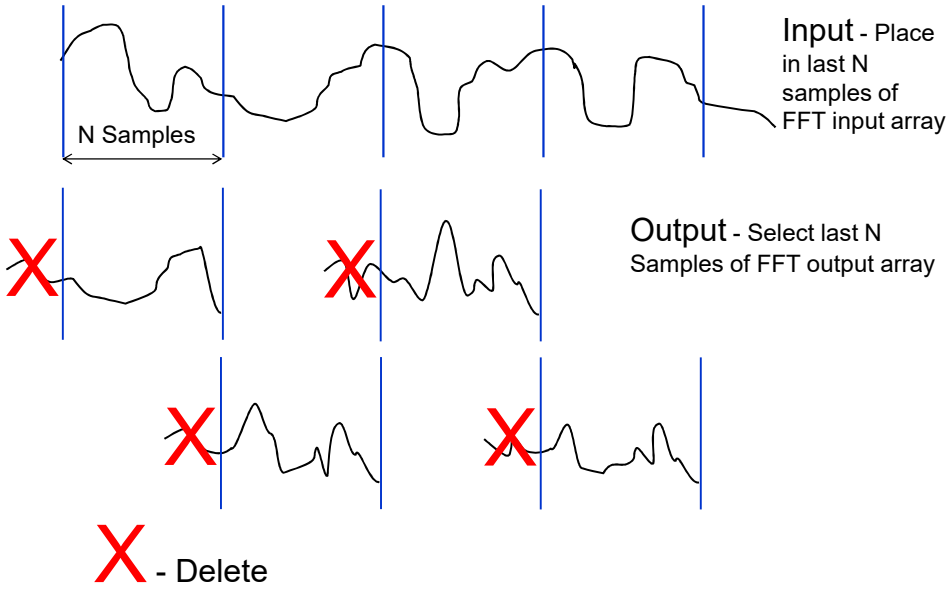
Overlap-add Implementation



C Code To Implement Frequency Domain Convolution (Overlap-add)

```
                                    /* Perform FFT on data */
SDA_rfft (SrcPtr, TempArrayPtr, FFTLength);

                                    /* Complex multiply */
SDA_complex_multiply_2 (SrcPtr, TempArrayPtr, RealFreqDomainCoeffs,
      ImagFreqDomainCoeffs, DstPtr, TempArrayPtr, FFTLength);

                                    /* Perform inverse FFT */
SDA_cifft (DstPtr, TempArrayPtr, FFTLength);

                  /* Add overlap for next iteration*/
SDA_add_2 (DstPtr, OverlapArrayPtr, DstPtr, (FilterLength-1));

                  /* Copy remainder of output for next iteration */
SDA_copy (DstPtr+DataLength, OverlapArrayPtr, (FilterLength-1));
```

Overlap-save Implementation



C Code To Implement Frequency Domain Convolution (Overlap-save)

```
                                    /* Copy source data + overlap */
SDA_copy (SrcPtr, OverlapArrayPtr+(FFTLength-DataLength), DataLength);

                                    /* Perform FFT on data */
SDA_rfft (OverlapArrayPtr, TempArrayPtr, FFTLength);

SDA_complex_multiply_2 (SrcPtr, TempArrayPtr, RealFreqDomainCoeffs,
      ImagFreqDomainCoeffs, DstPtr, TempArrayPtr, FFTLength);

                                    /* Perform inverse FFT */
SDA_cifft (DstPtr, TempArrayPtr, FFTLength);

                                    /* Copy and shift overlap from
                                    input array for next iteration */
SDA_copy (SrcPtr+DataLength-(FilterLength-1),
      OverlapArrayPtr+FFTLength-DataLength-(FilterLength-1),
      (FilterLength-1));
```

Frequency Domain Correlation