Chapter 16
Adaptive Filters

Learning Objectives

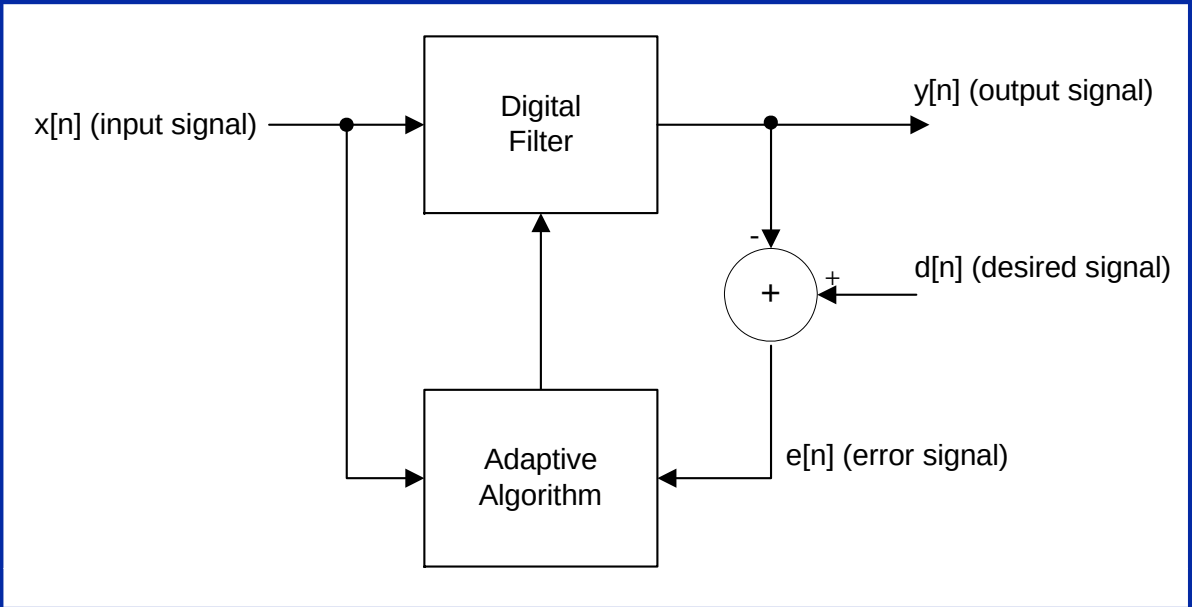
- Introduction to adaptive filtering.
- LMS update algorithm.
- Implementation of an adaptive filter using the LMS algorithm.

Introduction

- Adaptive filters differ from other filters such as FIR and IIR in the sense that:
  - The coefficients are not determined by a set of desired specifications.
  - The coefficients are not fixed.
- With adaptive filters the specifications are not known and change with time.
- Applications include: process control, medical instrumentation, speech processing, echo and noise calculation and channel equalisation.

Introduction

- To construct an adaptive filter the following selections have to be made:
  - Which method to use to update the coefficients of the selected filter.
  - Whether to use an FIR or IIR filter.



Introduction

- The real challenge for designing an adaptive filter resides with the adaptive algorithm.
- The algorithm needs to have the following properties:
  - Practical to implement.
  - Adapt the coefficients quickly.
  - Provide the desired performance.

The LMS Update Algorithm

- **The basic premise of the LMS algorithm is the use of the instantaneous estimates of the gradient in the steepest descent algorithm:**

$$h_n(k) = h_{n-1}(k) + \beta\Delta_{n,k}.$$

β = step size parameter

$\Delta_{n,k}$ = gradient vector that makes H(n) approach the optimal value H_{opt}

- **It has been shown that (Widrow and Stearns, 1985):**

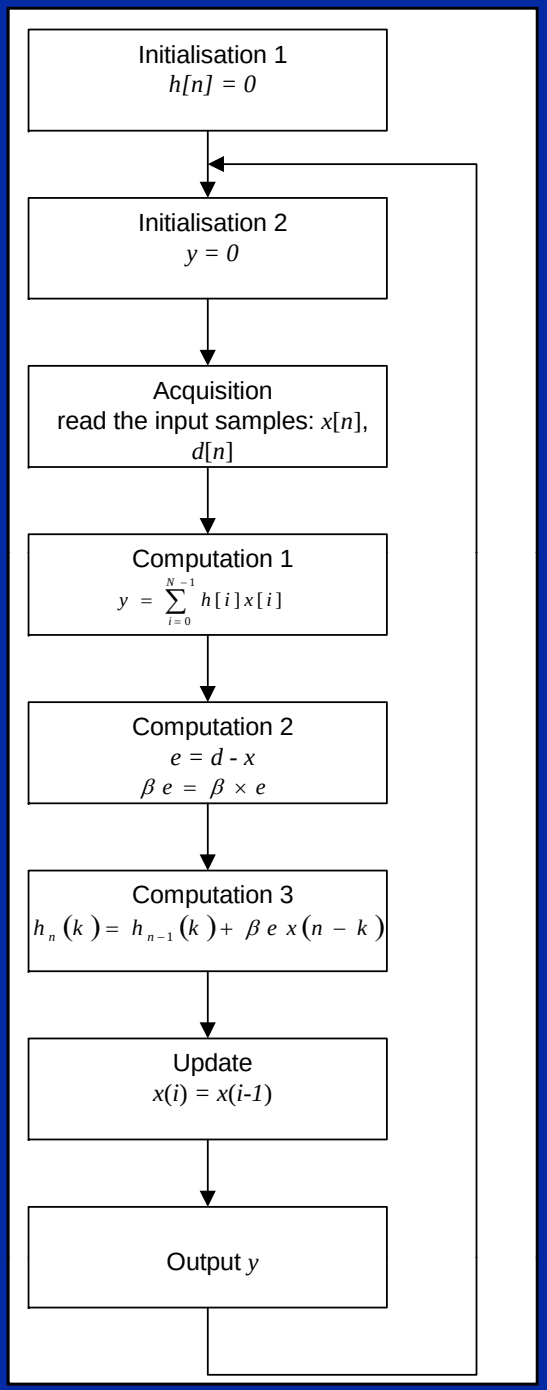
$$\Delta_{n,k} = e(n)x(n-k).$$

e(n) is the error signal, where: e(n) = d(n) - y(n)

- **Finally:**

$$h_n(k) = h_{n-1}(k) + \beta e(n)x(n-k).$$

LMS algorithm Implementation



LMS algorithm Implementation

```
temp = MCBSP0_DRR;                                  // Read new sample x(n)

X[0] = (short) temp;
D = X[0];                                           // Set desired equal to x(n) for this
                                                    // application
Y=0;

for(i=0;i<N;i++)
    Y = Y + ((_mpy(h[i],X[i])) << 1) ;              // Do the FIR filter

E = D -(short) (Y>>16);                             // Calculate the error
BETA_E =(short)((_mpy(beta,E)) >>15);               // Multiply error by step size parameter

for(i=N-1;i>=0;i--)
{
    h[i] = h[i] +((_mpy(BETA_E,X[i])) >> 15);       // Update filter coefficients
    X[i]=X[i-1];
}

MCBSP0_DXR = (temp &0xffff0000) | (((short)(Y>>16))&0x0000ffff);     // Write output
```

Adaptive Filter Code

- **Code location:**
  - **\Code\Chapter 16 - Adaptive Filter**
- **Projects:**
  - **Fixed Point in C: \Lms_C_Fixed**
  - **Floating Point in C: \Lms_C_Float**
  - **Fixed Point in Linear Asm: \Lms_Asm_Fixed**
- **Further reading:**
  - **Widrow and Stearns, 1985...**

Chapter 16
Adaptive Filters
- End -