Fixed point filter implementation

CPE 621 Advanced Microcomputer Techniques

Dr. Emil Jovanov

Digital filters

- Processing previous inputs (X[i]) and outputs (Y[i]) to evaluate the value of the current output sample
  - nb input samples and coefficients
  - na previous output samples and coefficients
  - current output is the sum of products of previous samples and coefficients
- Types of filters
  - IIR (Infinite Impulse Response) → general form
  - FIR (Finite Impulse Response) → only input samples X[i]
- Matlab function – filter
  - run Matlab "filtering" demo

```
FILTER One-dimensional digital filter.
   Y = FILTER(B,A,X) filters the data in vector X with the
   filter described by vectors A and B to create the filtered
   data Y.  The filter is a "Direct Form II Transposed"
   implementation of the standard difference equation:

   a(1)*y(n) = b(1)*x(n) + b(2)*x(n-1) + ... + b(nb+1)*x(n-nb)
                         - a(2)*y(n-1) - ... - a(na+1)*y(n-na)
```

Matlab filter Design and Analysis

- Matlab tools
  - sptool for preliminary signal processing and analysis
    - Example 3D accelerometer with LP and HP filtering
      - Signal viewer
      - Filter design and testing
      - Spectral analysis
  - fdtool for filter design and export
    - Tools → generate C header file
- Other tools …

Fixed point filter implementation

- Microcontrollers emulate floating point operations
  - Running fixed point operations much faster
  - The precision may not be sufficient for some applications
  - Example ffilt.c on our web-site
- Representing floating point numbers using fixed point values (arithmetic operations)
- Assume:
  - max(coefficient value) = MAX_INT
  - scale all coefficients to MAX_INT
- Optimize individual terms

Filter implementation - example

- Optimized processing
- Example - low pass IIR filter, coefficients:

```
b(1) -> 0.009236
b(2) -> 0.018472        a(2) -> -1.710329
b(3) -> 0.009236        a(3) -> 0.747274
```

  - filter coefficient: 0.009236
  - fixed point coef. value: 65536*()=605.290496 (0x25D)
  - binary value 0b 0000 0010 0101 1101 010010
  - Loosing 6-bits of precision!!!
  - Old value: 0b0000001001011101
  - New value: 0b1001011101010010 (dec. 38740)
- Processing
  - temporary result: unsigned long templ;
  - templ += (38740 * x[0]) >> 6;

fix_filt.c

```
/******
  file: fix_filt.c
  description: Fixed point FIR and IIR filtering routines,
               all procedures manually optimized for maximum precision.
  author: Emil Jovanov
  date: November 3, 2001. */

// input & output samples
#define FILT_LEN 12
int NB=FILT_LEN;                                        // filter length

/*** FIR filter - fixed point ***/
void xfir_filter(int * x, int * y, int sample) {
     /* fixed point filter procedure, xin - input signal ,  yout - filtered input signal */
               long templ;
               register int ii;

               /* the latest sample is at index 0, all other are shifted */
               for (ii=NB-1;ii>0;ii--) {
                         x[ii]=x[ii-1];
                         y[ii] = y[ii-1];
                         }

               x[0]=sample;

               /*** B coefficients */
               templ=0;
               templ += (44530 * x[0]) >> 7;          /* b(1) -> 0.005308 */
               templ += (47034 * x[1]) >> 5;          /* b(2) -> 0.022428 */
               templ += (58693 * x[2]) >> 4;          /* b(3) -> 0.055974 */
               templ += (53836 * x[3]) >> 3;          /* b(4) -> 0.102684 */
               templ += (39233 * x[4]) >> 2;          /* b(5) -> 0.149662 */
               templ += (47034 * x[5]) >> 2;          /* b(6) -> 0.179419 */
               templ += (47034 * x[6]) >> 2;          /* b(7) -> 0.179419 */
               templ += (39233 * x[7]) >> 2;          /* b(8) -> 0.149662 */
               templ += (53836 * x[8]) >> 3;          /* b(9) -> 0.102684 */
               templ += (58693 * x[9]) >> 4;          /* b(10) -> 0.055974 */
               templ += (47034 * x[10]) >> 5;         /* b(11) -> 0.022428 */
               templ += (44530 * x[11]) >> 7;         /* b(12) -> 0.005308 */

               y[0]=templ >> 16;
}

*** Filter initialization ***/
void filt_init_var(int *x, int *y) {
  register int ii;

  for (ii=0; ii<FILT_LEN; ii++)
    x[ii] = y[ii] = 0;
}

/*** IIR filter - fixed point ***/
void xiir_filter(int * x, int * y, int sample) {
               /* fixed point filter procedure
                         xin - input signal
                         yout - filtered input signal
               */
               long templ;
               register int ii;

               /* the latest sample is at index 0, all other are shifted */
               for (ii=NB-1;ii>0;ii--) {
                         x[ii]=x[ii-1];
                         y[ii]=y[ii-1];
               }
               x[0]=sample;

               /*** B coefficients */
               templ=0;
               templ += (38740 * x[0]) >> 6;          /* b(1) -> 0.009236 */
               templ += (38740 * x[1]) >> 5;          /* b(2) -> 0.018472 */
               templ += (38740 * x[2]) >> 6;          /* b(3) -> 0.009236 */
               /*** A coefficients */
               templ += (56044 * y[1]) << 1;          /* a(2) -> -1.710329 */
               templ -= (48973 * y[2]);               /* a(3) -> 0.747274 */

               y[0]=templ >> 16;
}
```

Filter Coefficients

- Group scaling of coefficients
  - Scale factor 1/max_coeff_value:
    - 1/1.710329 = 0.58 (1/2)
  - Coeff_shift = 0.5*65536
- Individual scaling of coefficients
  - Scale factor 1/coeff_value:
    - 1/0.005308 = 188
  - Coeff_shift = the largest power of two → 128 (2^7)
    - Coefficient value: round(128*65536*0.005308) = 44,527
- Temporary result:
  - templ += (sample * coeff) >> coeff_shift