EE 512

DIGITAL SIGNAL PROCESSING

Session 13

October 13, 1992

M. Azimi 800-525-4950, ext. 7956

Colorado State University

Problems

1) Since L equals to the minimum power of 2 greater than (M+N-1), the memory is not used in an efficient way.

2) If $N \ll L$ a waste of time results in the Computation of an L point FFT.

3) Often it is necessary to process a very long input sequence which requires Considerable amount of memory

Block Processing of FIR Filters

Sectioning Technique

Start here

The problem is usually solved by breaking up the input sequence into partitions that can be fitted into available primary memory. This method is refered to as "Sectioning method". The general advantages over the above method are

a) Reduced memory allocation

b) Reduced processing time (requires less I/O)

c) With appropriate choice of block sizes

the Computation time can be minimized.

There are two types of Sectioning techniques

1) Select-Save method 2) overlap add method.

1) Select-Save Method

This method involves constructing the filtered output by overlapping sections of the input. The degree of overlap between the sections is so chosen to eliminate the effect of wraparound error terms

Assume $\{x(m)\}$ is of size M and

$\{h(n)\}$ is of size N and $M \gg N$ e.g. $M = 1024$

$N = 10$. The select save algorithm can be organized in accordance with following steps

1. Choose the size of section D such as $D > N$. The proper choice of D is dissused later

2. Extend the impulse response to size D by adding an appropriate set of zeros. Compute the DFT of size D of the extended impulse

response sequence and store the result.

3. Read the first D points of the input sequence $\{\hat{x}(n)\}$, $n = 0, \cdots D-1$.

4. Compute the DFT of size D of input section Form the product of the DFT's of input section and that of the impulse response and take IDFT of the result. Call this $\hat{y}(K)$.

5. Part of $\hat{y}(K)$ contains wrap around error and should be discarded, this subsection is defined by the range of K

$$K = 0, 1, \cdots N-2$$

The valid part that will be retained are for range of K

$$K = N-1, \cdots D-1$$

the term "Select-Save arises from this operation.

In order to reduce the memory working space the desired part of output section of size $D-N+1$ can be placed into the same memory location as the first $D-N+1$ points of input section.

6. Form a new section $\{\hat{x}(n)\}$ from $x(n)$ of size D such that the first N-1 points of the new section are the same as last N-1 points of the old section. That is, point D- N+2 of the old section is now point 1 of the new section.

7. Repeat steps 4 through 6 until point N-1 of $x(n)$ is included in the new section. Add points of zeros to the new section if necessary to make it of size D and repeat 4 through 6.

At the Completion of step 7, the partial output sections that are placed together and saved, form the desired output.

2) Overlap - Add Method

Filtered sections are overlapped and added to Construct the output.

Steps 1 and 2 are the same as Select - Save method.

Step 3. Read a section of input of size D-N+1 and call this $\{\tilde{x}(n)\}$, the remainder of the section is filled with zeros.

4. Compute DFT of size D of the input section and form the

$$\tilde{y}(K) = \text{IDFT}\{ H(r)\, \tilde{X}(r) \}$$

5. The sequence $\tilde{y}(K)$ is added to the previous accumulation of $\tilde{y}(K)$. Adding takes place in the region of overlap i.e where N-1 zeros are added.
This is the origin of term overlap-add method

6. A new section is redefined by replacing the previously used section of size D-N+1 by the next subsection of the same size out of $x(n)$.

7. Repeat steps 4 through 6 until point N-1 of $x(n)$ is included in the newly formed section. Add zeros if necessary to make the last section of size D.

At the completion of step 7 the partial output results which are placed together form the desired output $y(k)$.

Remarks

a) Select Save method is found to be more efficient than the overlap-add method because

1) Although the total number of multiplications is the same for both, overlap add method requires more additions

2) Needs considerable book-keeping and storage for overlapping parts

3) An extra path through the output data file is required to perform the additions in the overlap region. Consequently I/O Complexity increases

4) May give rise to overflow due to the additions.

b) If the first N-1 points of the input contains important information that must be processed, the input must be enlarged by "padding" so that the discard operation of step 5 of (SSA) would not result in loss of valid information

$$x_e(n) = \begin{cases} c & \text{for} \quad 0 \leq n \leq N-2 \\ x(n) & N-1 \leq n \leq m+N-1 \end{cases}$$

where c is any constant.

Computational Time For Sectioning Technique

Sectioning algorithms require the partial overlapping of the points in the adjoining sections, the acceptable part of the sections which is processed in each iteration of the algorithm is a subsection of size $D - N + 1$. The the total number of blocks is

$$N_{blocks} = \left[\frac{M}{D-N+1} \right]$$

[a] stands rounding up operation to the next integer above the value of a.

The total number of complex multiplications required is

$$N_{total} = N_{blocks} (2N_{FFT} + D)$$

where N_{FFT} is the number of complex multiplications required to take a DFT of size D. The number of complex multiplications required to compute the DFT of impulse response is not included since this can be done beforehand and is negligible in comparison of the other DFT's.

Block Processing
2- IIR Filters

Implementation of Recursive Digital Filters

Using Block Processing Technique

Consider an LTI recursive (IIR) filter described by a 1-D difference equation

$$\sum_{i=0}^{N} b_i\, y(n-i) = \sum_{j=0}^{M} a_j\, x(n-j) \qquad (1)$$

This can be written as convolution operation i.e

$$b(k) * y(k) = a(k) * x(k) \qquad (2)$$

This can be arranged in matrix form

$$\begin{bmatrix} b_0 & 0 & 0 & - & - & - \\ b_1 & b_0 & 0 & - & - & 0 \\ \vdots & & & & & \\ b_N & - & - & - & - & b_0 \\ 0 & b_N & - & - & - & b_0 \\ & \ddots & & & & \ddots \end{bmatrix} \begin{bmatrix} y(0) \\ y(1) \\ \vdots \\ \vdots \end{bmatrix} =$$

$$\begin{bmatrix} a_0 & 0 & 0 & 0 & \cdots & 0 \\ a_1 & a_0 & 0 & \cdots & & 0 \\ \vdots & & & & & \\ a_m & \cdots & & a_0 & & \\ 0 & a_m & \cdots & a_0 & & \\ & \ddots & & & \ddots & \end{bmatrix} \begin{bmatrix} x(0) \\ x(1) \\ \vdots \\ \\ \\ \\ \\ \vdots \end{bmatrix} \qquad (3)$$

Partitioning the input and output in blocks of size L ($L \geq \max[m, N]$) with subscripts ordered lexicographically we get

$$X_0 = \begin{bmatrix} x(0) & x(1) & \cdots & x(L-1) \end{bmatrix}^T$$

$$X_1 = \begin{bmatrix} x(L) & x(L+1) & \cdots & x(2L-1) \end{bmatrix}^T$$

In general

$$X_i = \begin{bmatrix} x(iL) & x(iL+1) & \cdots & x((i+1)L-1) \end{bmatrix}^T \qquad (4)$$

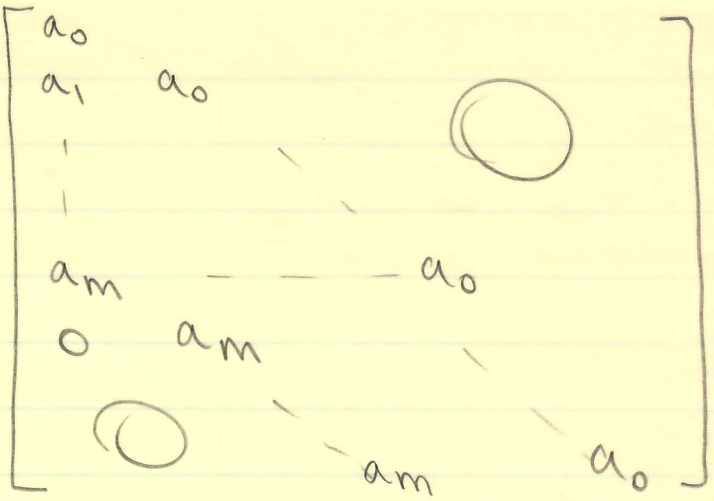
and similarly for Y_i we can partition the matrices and obtain

$$\begin{bmatrix} B_0 & & & \\ B_1 & B_0 & & \\ 0 & B_1 & B_0 & \\ & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} Y_0 \\ Y_1 \\ \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} A_0 & & & \\ A_1 & A_0 & & \\ 0 & A_1 & A_0 & \\ & \ddots & \ddots & \ddots \end{bmatrix} \begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ \vdots \\ \vdots \end{bmatrix} \quad (5)$$

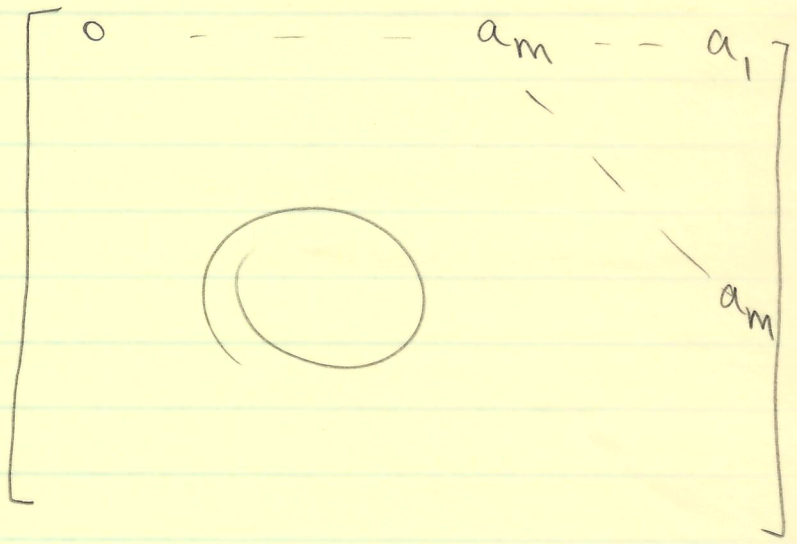
where $B_0 = \begin{bmatrix} b_0 & & & & \\ b_1 & b_0 & & & \\ \vdots & \ddots & \ddots & & \\ b_N & \cdots & \cdots & b_0 & \\ 0 & b_N & \cdots & \cdots & b_0 \\ & \ddots & \ddots & & \\ & & b_N & & \end{bmatrix}$

$$B_1 = \begin{bmatrix} 0 & \cdots & b_N & \cdots & b_1 \\ 0 & \cdots & \cdots & b_N & b_2 \\ & & & \ddots & \\ & & & & b_N \\ & & & & \\ & & & & \end{bmatrix}$$

Similarly $A_0 =$



and $A_1 =$



From Equation 5 we get (6)

$$B_0 Y_{i+1} + B_1 Y_i = A_0 X_{i+1} + A_1 X_i$$

or

$$Y_{i+1} = -B_0^{-1} B_1 Y_i + B_0^{-1} A_0 X_{i+1} + B_0^{-1} A_1 X_i$$

or simply

$$Y_i = C Y_{i-1} + D_0 X_i + D_1 X_{i-1}$$

This equation is called "block recursive equation".

Remarks

1) Using this equation, each output block can recursively be obtained from the past output block and the past and present input blocks.

2) If the block dimension is reduced to 1 the block recursive equation becomes the original 1-D difference equation

3) matrices involved in this structure are all Toeplitz (either lower or upper triangular). This makes it possible to perform the matrix-vector operation that are convolution operators by fast transform technique such as FFT, NTT and PT.

4) Matrix B_0^{-1} is also a lower triangular Toeplitz since B_0 is a lower triangular Toeplitz the inverse always exists. This matrix may be expressed in terms of the truncated impulse response elements of the associated all-pole filter.

Special cases

1) 1-D Recursive All-pole filter

$$a_0 = 1 \quad \text{and} \quad a_i = 0 \quad i \neq 0$$

Then $A_0 = I_L$ and $A_1 = 0$

Thus $$Y_i = C\,Y_{i-1} + B_0^{-1} X_i$$

2) 1-D Non-recursive Digital filter

$$b_0 = 1 \quad \text{and} \quad b_i = 0$$

$$B_0 = I_L \quad \text{and} \quad B_1$$

Thus $$Y_i = A_0 X_i + A_1 X_{i-1}$$

This gives the same result as the overlap add method.

Computational Cost

Assume $m=N$ and $L \geq m$, efficiency will be achieved if some of the operations are carried out by the use of the FFT.

1) B_0^{-1} represents L-length linear Convolution which may be carried out by 2L-circular Convolution via FFT.

2) A_0 requires an L+M-length FFT Convolution or $(m+1)L - (m+1)\frac{m}{2}$ direct multiplications (if direct method is used)

3) A_1 and B_1 require 2M-length FFT Convolution or $(m+1)m/2$ direct multiplication

Note:

Rectangular operator $[A_1\ A_0]$ can be performed by an L+M-length FFT Convolution or $(m+1)L$ direct multiplications

If a very efficient FFT implementation is used for real-valued data then the total number of real multiplications required is

$L = 2^s$

$$N_{total} = 5\left[2L\, N_{FFT}(s)\right] + 6L$$

where

$$N_{FFT}(s) = 2^s(s-3) + 4$$

Direct method requires

$$N_{DMVP} = (M+1)\left[L + M/2\right] + \frac{L(L+1)}{2}$$