Chapter 4

Sliding Discrete Fractional Transforms
4.1 Introduction

Signal processing techniques are appreciated when they can be implemented for real time signal processing applications. In real time applications, samples of input signal arrive in a sequential manner. Irrespective of the computational technique and hardware arrangement, a block of \( N \) input samples is processed at a time while computing a transform. For the computation of transform of next block of \( N \) input samples, the system has to wait for the arrival of all the \( N \) new input samples. However, it is possible to continue with the processing as and when a new sample arrives by processing the newly arrived sample, with the rest of the \( N-1 \) samples being the past samples. Each time, the computation window slides by one sample. This technique is referred to as sliding technique. This is employed in the computation of discrete Fourier transform (DFT) for real time signals and is referred to as sliding DFT (SDFT) /1,2/. Recursive computation of coefficients is achieved by utilizing the already computed coefficients of the previous block and it reduces the number of computations involved. The method would be ideal for fast computation of the transform when the processing time is less than the time for arrival of the required number of samples. The processing system need not wait for the samples to accumulate.
Currently available different discretization techniques of fractional Fourier transform (FRFT) have led to different hardware implementation for the computation of discrete FRFT (DFRFT). However, some of the techniques are implemented with multi-input-multi-output systems /3,4/ and some are multirate systems /5-9/. In all these implementations, the hardware for the computation /9/ typically consist of three sections: a \( N \)-fold interpolation section, the transform kernel section that transforms an array of input samples into a transformed vector, and a \( N \)-fold decimation section. Amongst these, interpolation and decimation are multirate operations. The sequential input samples are converted to parallel form after interpolation, and converted back to sequential form before decimation, so that at the output, all the coefficients are obtained simultaneously. With the increase in the number of samples, filter bank approach can save a large amount of time in computation with the expense of hardware. Computation of discrete fractional cosine transform (DFRCT) and discrete fractional sine transform (DFRST) also involve multirate operations. While DFRCT can be computed as the DFRFT of symmetrically extended signal, DFRST can be computed as the DFRFT of antisymmetrically extended signal.

In this chapter, the methods of computing three sliding discrete fractional transforms: sliding discrete fractional Fourier transform (SDFRFT), sliding discrete fractional cosine transform (SDFRCT) and sliding discrete fractional sine transform (SDFRST) are presented. Their performances are
compared in terms of computational complexity, variance of quantization error and signal to noise ratio. For comparing the performances of each of the proposed sliding discrete fractional transforms with SDFT, signal-to-noise ratio has been calculated for one of the signals in sound quality assessment material (SQAM). When a particular time-frequency bin is to be observed, computational complexity in the case of sliding discrete fractional transform is less than that in discrete fractional transform. Sliding discrete fractional transforms require reduced number of bits for the coefficient representation in comparison with SDFT. Sliding discrete fractional sine transform performs better in comparison with sliding discrete fractional Fourier and sliding discrete fractional cosine transforms.

### 4.2 Sliding discrete fractional transforms and their Implementations

Based on the definitions of the three discrete fractional transforms, a generalized kernel derived in chapter 2 is reproduced here for convenience.

\[
Y_{Ga}(m) = \sum_{n=1}^{U} F_{Ga}(m,n)y(n)
\]  

with the limits of the summation index being \( L = -N \) and \( U = N \) for DFRFT, \( L = 0 \) and \( U = P \) for DFRCT and \( L = 1 \) and \( U = P - 1 \) for DFRST. The suffix \( G \) distinguishes the three transforms. \( G = f, c \) and \( s \) represent Fourier, cosine and sine transforms, respectively. The kernel, \( F_{Ga}(m,n) \), can be expressed as
\[ F_{G^a}(m,n) = e^{\frac{\cot a}{2} m^2 \delta w^2 + n^2 \delta w^2} K_G(m,n). \] (2.68)

The term \( K_G(m,n) \) is different for different fractional transforms. The additional suffix \( G \) is used for distinguishing the three transforms.

\[ K_G(m,n) = A_{fa} e^{-j \frac{\delta m \delta n}{p}} \text{ for } G = f \]
\[ = A_{ca} k_m k_n \cos\left(\frac{\pi m n}{p}\right) \text{ for } G = c \text{ and } \]
\[ = A_{sa} \sin\left(\frac{\pi m n}{p}\right) \text{ for } G = s. \] (2.69)

In the computation of the sliding transform, the process performed on \( p-i \) samples of the previous block of \( p \) samples is required while considering them along with the new sample. The closed form of the transform is useful in performing additional processing on \( p-i \) samples of the previous block. Consider a sequence of input samples

\[ \ldots y(n-2), y(n-1), y(n), y(n+1), y(n+2), y(n+3) \ldots \]

Using Eqn. (2.67), discrete fractional transform of \( p \) input samples is given by

\[ Y^G_{A^a}(m) = \sum_{k=0}^{U} e^{\frac{\cot a}{2} m^2 \delta w^2 + n^2 \delta w^2} K_G(m,k)y(n-1-k). \]

\[ = \sum_{k=0}^{U} e^{\frac{\cot a}{2} m^2 \delta w^2} K_G(m,k)y(n-1-k) \]
\[ = e^{\frac{\cot a}{2} m^2 \delta w^2} (y(n-1-L)e^{\frac{\cot a}{2} (L+1)^2 \delta w^2} K_G(m,L) + y(n-1-(L+1))e^{\frac{\cot a}{2} (U-1)^2 \delta w^2} K_G(m,U-1) + y(n-1-U)e^{\frac{\cot a}{2} U^2 \delta w^2} K_G(m,U)). \] (4.1)
Sliding the window by one sample to the right, discrete fractional transform
of another \( P \) samples is given by

\[
Y_{a,n}^G(m) = \sum_{k=L}^{U} e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} n k} K_G(m,k) y(n-k).
\]

\[
= \sum_{k=L}^{U} e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} m^2 \omega_1^2 + k^2 \omega_2^2} K_G(m,k) y(n-k)
\]

\[
= e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} m^2 \omega_1^2} (y(n-L)) e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} \omega_1^2} K_G(m,L)
\]

\[
+ y(n-(L+1)) e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} (L+1)^2 \omega_1^2} K_G(m,L+1) ... + y(n-(U-1)) e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} (U-1)^2 \omega_1^2} K_G(m,U-1)
\]

\[
+ y(n-U) e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} U^2 \omega_1^2} K_G(m,U)).
\]

Comparing (4.1) and (4.2), all the samples involved in computation of
\( Y_{a,n-U}(m) \) are also in \( Y_{a,n}(m) \) except for the last sample, \( y(n-1-U) \). There is
one new sample, \( y(n-L) \), in \( Y_{a,n}(m) \) as the first sample. The multiplication
factor for \( y(n-1-L) \) in \( Y_{a,n}(m) \) is \( e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} (L+1)^2 \omega_1^2} K_G(m,L+1) \) and that in \( Y_{a,n-1}(m) \)
is \( e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} \omega_1^2} K_G(m,L) \). Therefore change in multiplication factor for \( y(n-1-L) \) is

\[
e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} (L+1)^2 \omega_1^2} K_G(m,L+1)/ (e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} \omega_1^2} K_G(m,L)).
\]

Similarly for \( y(n-L-2) \) in \( Y_{a,n}(m) \) the multiplication factor is \( e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} (L+2)^2 \omega_1^2} K_G(m,L+2) \) and that in \( Y_{a,n-1}(m) \) is

\[
e^{\frac{j \cot \alpha \omega_1 \omega_2}{2} \omega_1^2} K_G(m,L+1). \]

Change in multiplication factor for \( y(n-L-2) \) is
Similarly, change in multiplication factor for all the terms can be determined.

A valid output of sliding discrete fractional transform will be available only after processing the first \( p \) input samples. Subsequent output will be available soon after the arrival of new input sample. The sliding discrete fractional transforms cannot be recursive, as in the case of SDFT \( /1,2/\), because of the chirp multiplication in fractional transforms. The implementation of sliding discrete fractional transforms is shown in fig. 4.1 for \( p = 5 \).

Fig. 4.1: Implementation of sliding discrete fractional transform for \( P=5 \).
The multiplication factors are different for different fractional transforms. Table 4.1 shows the multiplication factors for the three sliding discrete fractional transforms. $A_G$ to $H_G$ are as in Table 4.1 for $G = f, c$ or $s$.

Only in the case of SDFRCT, $A_G$ is real. As the summation index in the case of SDFRCT is 0 to $p$, there is an additional term in its computation in comparison with SDFRFT. Similarly, summation index in the case of SDFRST is from 1 to $p-1$ and number of terms is one less in comparison with SDFRFT.

Table 4.1. Multiplication factors for the sliding discrete fractional transform when implemented with $P=5$.

<table>
<thead>
<tr>
<th>Multiplication factor</th>
<th>SDFRFT</th>
<th>SDFRCT</th>
<th>SDFRST</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A_G$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\frac{1}{\sqrt{2}}$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}}$</td>
</tr>
<tr>
<td>$B_G$</td>
<td>$\frac{-\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\sqrt{2} e^{-\frac{\alpha}{2}} \cos(m\pi/5)$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \sin(2m\pi/5) / \sin(m\pi/5)$</td>
</tr>
<tr>
<td>$C_G$</td>
<td>$\frac{-\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \cos(2m\pi/5) / \cos(m\pi/5)$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \sin(3m\pi/5) / \sin(2m\pi/5)$</td>
</tr>
<tr>
<td>$D_G$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \cos(3m\pi/5) / \cos(2m\pi/5)$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \sin(4m\pi/5) / \sin(3m\pi/5)$</td>
</tr>
<tr>
<td>$E_G$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \cos(4m\pi/5) / \cos(3m\pi/5)$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \sin(5m\pi/5) / \sin(4m\pi/5)$</td>
</tr>
<tr>
<td>$F_G$</td>
<td>$\frac{-\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} e^{-\frac{\sin(m\pi/5)}{5}}$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \cos(5m\pi/5) / \cos(4m\pi/5)$</td>
<td>$\frac{\cot\alpha}{2} \frac{\sin(m\pi/5)}{e^{-\frac{\alpha}{2}}} \sin(6m\pi/5) / \sin(5m\pi/5)$</td>
</tr>
<tr>
<td>$H_G$</td>
<td>$A_{fa} e^{-\frac{\alpha}{2} m^2 \omega t}$</td>
<td>$A_{ca} k_{m} e^{-\frac{\alpha}{2} m^2 \omega t}$</td>
<td>$A_{ca} e^{-\frac{\alpha}{2} m^2 \omega t}$</td>
</tr>
</tbody>
</table>
4.3 Performance evaluation of sliding discrete fractional transforms

The advantage of using fractional transforms can easily be observed by calculating normalized square magnitude of the transform coefficients for a chirp signal [3] such as shown in fig. 4.2. Figure 4.3 shows the normalized square magnitude of fractional transforms obtained using DFRFT (fig. 4.3(a)), DFRCT (fig. 4.3(b)) and DFRST (fig. 4.3(c)) and the normalized square magnitude obtained using ordinary transforms, DFT (fig. 4.3(d)), discrete cosine transform (DCT) (fig. 4.3(e)) and discrete sine transform (DST) (fig. 4.3(f)). The value of order parameter $\alpha$ for compact domain is 0.58, 1.3 and 0.78, respectively, for DFRFT, DFRCT and DFRST. The spread of the signal in the transform domain, in all the three ordinary transforms, is wider in comparison with those in fractional transforms. It is evident that the number of coefficients, which are significant and required for representing the signal, is less in fractional transform domain. This feature of fractional transforms will enhance performance of their sliding versions.

Fig. 4.2 Chirp signal
It is well known that each complex multiplication requires 3 real multiplications /10/. In the case of SDFRFT, for a real input sequence, multiplication of the first term with two complex numbers requires 5 real multiplications. Each of the \( p \)-delayed versions of the input is multiplied by two complex numbers requiring \( 6(p-1) \) real multiplications. The sum is multiplied by a chirp signal and scaling factor, \( A_f \), requiring another 6 real multiplications. These operations lead to a total of \( 6p+5 \) real multiplications. In the case of SDFRCT and SDFRST, total number of real multiplications is \( 6p+7 \) and \( 6p-2 \), respectively. In this method, time required for computing the
transform is less due to minimization of number of multiplications per coefficient.

Computation of all the $p$ sliding transform coefficients is performed by parallel implementation of $p$ number of units shown in Fig. 4.1, with one unit for each of the coefficient. Such an arrangement requires $P$ times the number of real multipliers required per coefficient. Recently, because of the development of large-scale integrated circuits, emphasis on minimizing the number of multipliers has reduced. Many parallel devices are affordable rather than a few high-speed devices /7/. For real time applications, wherein samples arrive in a sequential manner, all the sliding discrete fractional transform coefficients are computed nearly at the same rate of arrival of input samples.

In the case of sliding discrete fractional transform, after an initial waiting period for the $p$ input samples to arrive, output is available at the rate of $T = \max(T_s, T_r)$. Here, $T_r$ is the time gap between the arrival of input samples and $T_s$ is the time taken for the computation of sliding discrete fractional transform. In the case of discrete fractional transform, $p$ point discrete fractional transform is obtained after $T_0 = T_p + PT_r$, where $T_p$ is the time taken for the computation of $p$ point discrete fractional transform. The term $PT_r$ represents the waiting time for the arrival of $p$ input samples. As the number of multiplications is less in sliding discrete fractional transform, $T_r \ll T_p$ and hence $T \ll T_0$. Waiting time for the arrival of $p$ input samples
can be overlapped with the computation of $P$ point discrete fractional transform, requiring $\max(T_p, PT_0)$ time to compute $P$ point discrete fractional transform. Sliding discrete fractional transform outperforms the computation of discrete fractional transform as $T << \max(T_p, PT_0)$.

The performance of any transform is evaluated by considering number of bits required for representing the real value. If $b$ bits is required for a real value, in fixed-point arithmetic, every real multiplication results in maximum of $2b$ bits. The product if rounded from $2b$ bits to $b$ bits leads to quantization error. This quantization error has a variance given by $/10-12/

$$\sigma_e^2 = \frac{2^{-2b}}{12}.$$  

(4.3)

Since $6P+5$ real multiplications are required in computing SDFRFT, the variance of quantization error is

$$\sigma_x^2 = \frac{2^{-2b}}{12} (6P+5).$$  

(4.4)

Similarly, in the computation of SDFRCT and SDFRST the variance of quantization error can be shown to be

$$\sigma_e^2 = \frac{2^{-2b}}{12} (6P+7)$$

and

$$\sigma_x^2 = \frac{2^{-2b}}{12} (6P-2), \text{ respectively.}$$  

(4.5)

Letting $2N = 2^n$, and $P \approx 2N$ for large $N$, in (4.4) and (4.5),

$$\sigma_y^2 = \sigma_z^2 = \sigma_x^2 = 2^{-2b+n-1}.$$  

(4.6)
Signal-to-noise-ratio (SNR) is defined as

\[
SNR_{G} = 10 \log_{10} \frac{\sigma_{G}^{2}}{\sigma_{F}^{2}}, G = f, c, s, \tag{4.7}
\]

with \( \sigma_{G}^{2} \) being the variance of the coefficients of the three sliding discrete fractional transforms. Table 4.2 lists the performance measures of the three sliding discrete fractional transforms for single coefficient. Number of multipliers required in the case of SDFRST is less in comparison with that in SDFRFT and SDFRCT.

Table 4.2: Performance measures of sliding discrete fractional transforms for single coefficient.

<table>
<thead>
<tr>
<th>Performance Measures</th>
<th>SDFRFT</th>
<th>SDFRCT</th>
<th>SDFRST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of real multiplications</td>
<td>( 6P + 5 )</td>
<td>( 6P + 7 )</td>
<td>( 6P - 2 )</td>
</tr>
<tr>
<td>Variance of quantization error</td>
<td>( 2^{-2b+n-1} )</td>
<td>( 2^{-2b+n-1} )</td>
<td>( 2^{-2b+n-1} )</td>
</tr>
<tr>
<td>SNR</td>
<td>( \sigma_{F}^{2} / 2^{-2b+n-1} )</td>
<td>( \sigma_{c}^{2} / 2^{-2b+n-1} )</td>
<td>( \sigma_{r}^{2} / 2^{-2b+n-1} )</td>
</tr>
</tbody>
</table>

Samples of audio signal X1, from sound quality assessment material (SQAM) shown in fig. 4.4, are considered for verifying above results. The sliding discrete fractional transform coefficients can be computed with and without restriction on the number of bits in each multiplication operation. The coefficients obtained are treated as practical values when computed with restriction on number of bits. The coefficients computed without restriction on the number of bits are treated as ideal values. Practical value
of SNR is computed as the ratio of sum of square of the practical values of
coefficients to the sum of square of the difference between the practical and
ideal values of coefficients. Ideal value of SNR is shown as a function of
number of bits in fig. 4.5 for all the three sliding discrete fractional
transforms with $P=5$. The SNR in all the three sliding discrete fractional
transforms are nearly same. Practical values of SNR obtained for all the
three sliding discrete fractional transforms and for SDFT are shown as a
function of number of bits in fig. 4.6. SNR is 15dB, 10dB and 30dB for
SDFRFT, SDFRCT and SDFRST, respectively, with 8 bits for representing
the coefficients. By increasing the number of bits by two fold, SNR is 65dB,
58dB and 78dB respectively, for SDFRFT, SDFRCT and SDFRST. With 8
bits for representing the coefficients, ideal values of SNR is 45dB and with
16 bits it is 93 dB. We find that SDFRST has better SNR in comparison
with SDFRFT or SDFRCT. SDFT has better SNR values in comparison
with the sliding discrete fractional transforms. In the case of SDFT, SNR
with 8 bits for representing the coefficients is 36dB and with 16 bits it is
85dB. The sliding discrete fractional transforms require 10-13 bits for
representing the coefficients with SNR of 30-40 dB, whereas SDFT requires
7-9 bits. The difference in the number of bits may be ascribed to the time-
frequency information available in the coefficients of sliding discrete
fractional transforms. For a given SNR, more bits are required to represent
all the coefficients in SDFT, in comparison with the bits required for
representing the coefficients obtained in a fractional transform. Therefore fractional transforms may be preferred over the ordinary transforms.

Fig. 4.4. Samples of $X_1$ from SQAM.

Fig. 4.5. Ideal SNR for SDFRFT (solid line), SDFRCT (plus sign line), SDFRST (dashed line)
4.4 Conclusions

Methods of implementing sliding discrete fractional transforms are presented. Their performances with regard to computational complexity, variance of quantization error and signal to noise ratio are compared. Because of reduced number of multiplications, sliding discrete fractional transforms are faster and simpler to implement. They are useful for real time practical applications when a particular time-frequency bin is to be analyzed. Computational complexity and the variance of quantization error are less in the case of sliding discrete fractional sine transform. In comparison with SDFT, sliding discrete fractional transforms require less number of coefficients to represent the signal and so also the total number of bits.
REFERENCES


/7/ Der-Feng Huang, Bor-Sen Chen, Proc. of Int. Conf. ASSP, 3, 1709, (1999).


/9/ Der-Feng Huang and Bor-Sen Chen, Signal Processing, 80, 1501, (2000).

