Speed testing of Sliding spectrum analysis

Emil Frenski, Member, IEEE
Dimitar Manolev
South-West University "Neofit Rilski", Blagoevgrad, Bulgaria

***Abstract**: The standard method for spectrum analysis in DSP is the Discrete Fourier transform (DFT), typically implemented using a Fast Fourier transform (FFT) algorithm. The reconstruction of the time-domain signal is then performed by the IFFT (Inverse Fast Fourier transform) algorithm. The FFT calculates the spectral components in a window, on a block-by-block basis. If that window is move by one sample, it is obvious that most of the information will remain the same. This article shows how to measure execution time of scripts realizing SDFT algorithm written for MatLab.*

***Keywords:** Sliding Discrete Fourier transform (SDFT), Fast Fourier transform (FFT), MatLab.*

1. INTRODUCTION

The discrete Fourier transform (DFT) plays very important role in the implementation of discrete-time signal-processing. The DFT is identical to samples of the Fourier transform. Computation of the *N*-point DFT corresponds to the computation of the Fourier transform at *N* equally spaced frequencies $\omega_k = 2\pi k / N$ (bins[1]), i.e. at *N* points on the unit circle in the z-plane. The DFT of finite – length sequence is (1)

$$S^k = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \tag{1}$$

where S^k represent the k-th frequency point (bin), $x[n]$ is sampled input data of size *N* and $W_N = e^{-j2\pi/N}$.

Since in (1), both $x[n]$ and S^k may be complex, *N* complex multiplication and (*N*-1) complex addition are required. Computational complexity of

[1] Frequency bins k corresponds to the band of frequencies centred at $\omega_k = 2\pi k / N$ with a bandwidth of approximately $2\pi / N$

each successive *N*-point output is approximate $O(N^2)$ where *O*(·) denotes order of. It is evident that the number of arithmetic operations required computing DFT becomes very large for large values of *N*.

Set of algorithms known as the fast Fourier transform (FFT) is used for reducing the computational complexity to $O(N \log_2 N)$.

The FFT is "fast" when all the *N* values of S^k are needed and the number of samples N is a power of two. But if we are only interested in the k-th value of the DFT, we have to compute the entire DFT - sequence and discard the unwanted values.

On the other hand, in the case of DFT, it has been noted that the algorithm can be implemented with $O(N)$ complexity, for any (non power of two) value of *N*. This can be achieved by using a set of parallel recursive digital filters [1] - this is called sliding DFT (SDFT). In the current literature, the term running DFT has also been used for this purpose.

2. SLIDING DISCRETE FOURIER TRANSFORM

The sliding DFT (SDFT) use the circular shift property. We use this shift principle to express sliding DFT process as [2][3][4]

$$S_{[n]}^k = \left[S_{[n-1]}^k - x[n-M] + x[n] \right] W_M^k, \tag{2}$$

where $S_{[n]}^k$ is the new spectral component at the time index [n] and $S_{[n-1]}^k$ is the previous spectral component at the time index [n-1]. The superscript *k* is associated with the *k*-th DFT bin. There is no requirement for the window size to be a power of two and we use *M* for sample data points, instead of the usual convention *N*. The difference between the current sample $x[n]$ and the last sample $x[n-M]$ can be computed once for each $S_{[n]}^k$.

The sliding algorithm (2) performs an M=16 point DFT on time samples is depicted in Fig. 1.

First the SDFT computes the DFT at the time index [n-1], second at the time index [n]. For this we forget the oldest sample $x[n-M-1]$ and accept new sample $x[n]$.

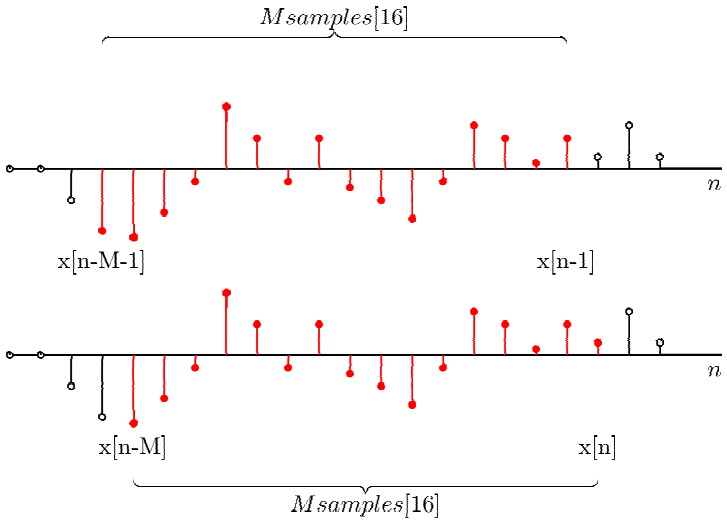

Fig. 1: Data samples at the time index [n] and [n-1].

The z transform which corresponds to a (2) is

$$S_{[n]}^{k} = \left[S_{[n]}^{k} z^{-1} - x[n] z^{-M} + x[n] \right] W_M^k . \tag{3}$$

And transfer function is

$$H(z) = \frac{\left(1 - z^{-M}\right) W_M^k}{1 - z^{-1} W_M^k} . \tag{4}$$

The transfer function (4) has *M* zeros located at the *M* root of 1 and single pole located at $z_1 = e^{(j2\pi/M)k}$. We see that the single pole cancelled the k-th zero (i.e.) transfer function having (M-1) zeros and no poles. As example, Fig 2. shows the zero – pole plot and frequency response for the *M*=16 and *k*=2

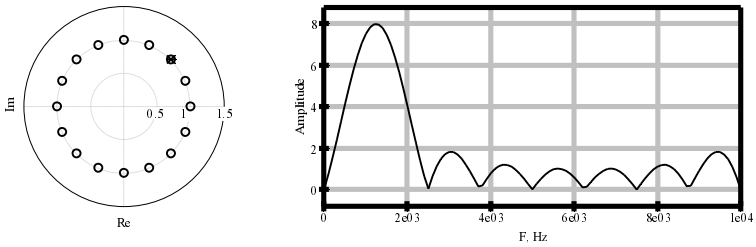

Fig. 2: Zero-pole plot and frequency response for the *M*=16 and *k*=2.

Equation (4) leads for filter structure shown in Fig. 3. The single - bin SDFT algorithm is implemented as an IIR filter with a comb filter followed by a complex resonator [2].

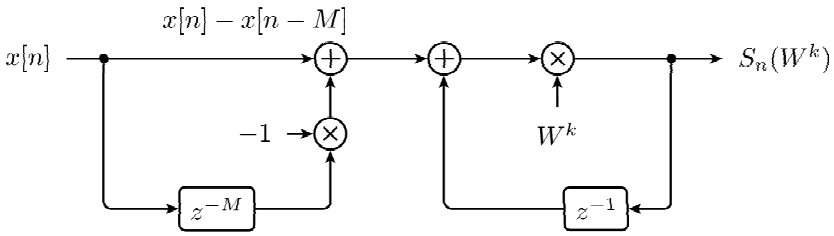

Fig. 3: The single - bin SDFT filter.

The frequency response of comb filter, complex resonator and single-bin SDFT filter is depicted in Fig. 4.

If we want to compute all M DFT spectral components, M resonators will be needed, all driven by a single comb filter. The comb filter delay of M samples forces the filter's transient response to be M–1 samples in length, so the output will not reach steady state until the Sk(n) sample.

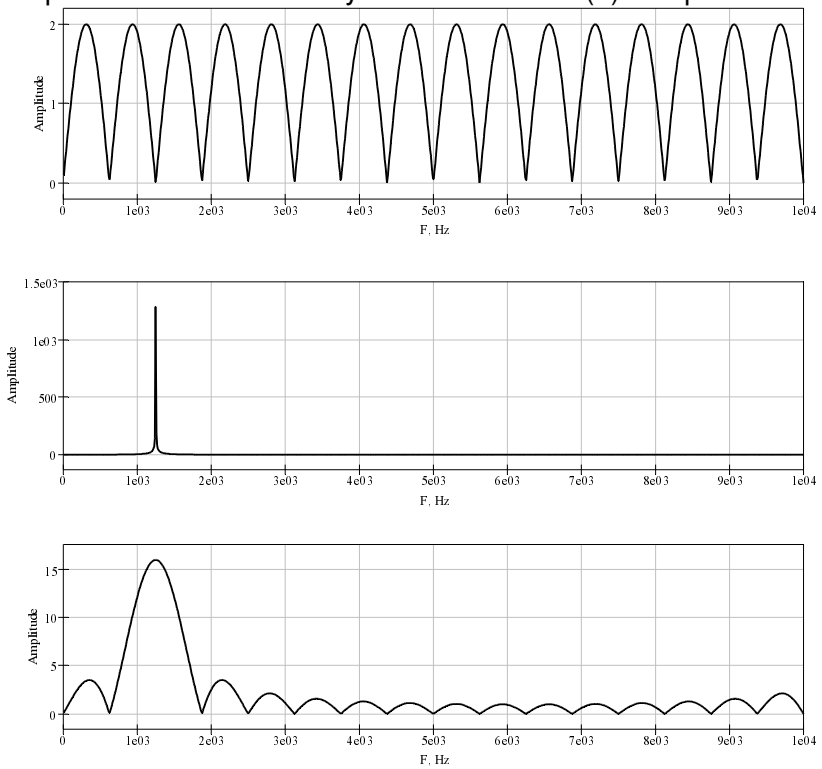

Fig. 4: The single - bin SDFT filter frequency response.

3. EXECUTION TIME IN MATLAB

The scripts performing a SDFT algorithm are written in MatLab. When calculating the execution time, we does not includes scripts realizing input-output operations, as well as those for drawing diagrams. Execution time is measured only on scripts realizing SDFT and SIDFT.

There are two ways to measure the execution time of the scripts in MatLab. First, by using the *tic* and *toc* functions, which use clock time. Second, through the functions *clock* and *etime*. The function *clock* uses the system time, changing periodically. Because of this, there can be no precise criteria for comparing the execution times. To calculate the execution time, we have used two different ways:

```
 % Using tic and toc functions
iStart = tic;
% Here is scripts of SDFT and SIDFT calculation
iElapsedTime = toc(iStart);
```

and

```
% Using clock and eTime  functions
iStart = clock;
% Here is scripts of SDFT and SIDFT calculation
iElapsedTime = etime(clock, iStart)
```

The results is shown in Fig. 5.

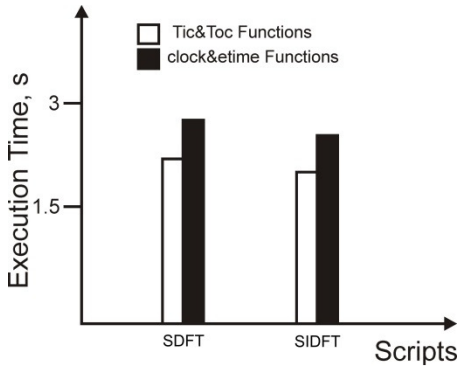

Fig. 5: Execution Times of the SDFT algorithm in MatLab.

4. CONCLUSIONS

Many factors influence the execution time of a task in the field of signal processing. In this paper we used Matlab version 7.12 on Windows 7 x64,

PC with processor Intel Celeron B800 / 4Gb RAM. Methods that are used to measure execution time is with ensure accuracy up to 0.1 seconds [5]. In this case used two different ways to measure the execution time on the same platform. Nevertheless, the execution times are different. Future work includes measuring the execution time of the SDFT algorithm on different platforms using advanced techniques.

5. REFERENCES

[1] Rabiner, L., Gold, B. (1975) Theory and Application of Digital Signal Processing. Upper Saddle River, NJ: Prentice Hall, pp. 382-383.
[2] Jacobsen, E., Lyons, R. (2003) The sliding DFT. IEEE Signal Processing Magazine, vol. 20, no. 2, pp. 74–80.
[3] Jacobsen, E., Lyons, R. (2004) An update to the sliding DFT. IEEE Signal Processing Magazine, vol. 21, no. 1, pp. 110–111.
[4] Farhang - Boroujeny, B., Lim Y. C. (1992) A comment on the computational complexity of sliding FFT. IEEE Transactions on Circuits and Systems, vol. 39, no. 12, pp 875 876.
[5] Martin K., McKeeman B. (2011) Accelerating the pace of Engineering and science. MathWorks,Inc.