Krzysztof Duda

Accurate, Guaranteed Stable, Sliding Discrete Fourier Transform

"DSP Tips and Tricks" introduces practical design and implementation signal processing algorithms that you may wish to incorporate into your designs. We welcome readers to submit their contributions. Contact Associate Editors Rick Lyons (R.Lyons@ieee.org) or C. Britton Rorabaugh (dspboss@aol.com).

The sliding discrete Fourier transform (SDFT) is a recursive algorithm that computes a DFT on a sample-by-sample basis. The SDFT is computationally efficient, but it suffers from accumulated errors and potential instabilities. As we shall see, previous SDFT algorithms described in the literature compromise results accuracy for guaranteed stability [1]–[3].

This article describes a new SDFT algorithm that is guaranteed stable without sacrificing accuracy—an algorithm we call the modulated SDFT (mSDFT).

Our algorithm is based on the DFT modulation property; for the chosen DFT bin with index k we use this property to effectively shift that bin to the position $k = 0$. Next, we simply compute the running sum in the sliding window of length N. This approach allows us to exclude the complex twiddle factor from the feedback in the resonator and avoid accumulated errors and potential instabilities.

While the details of traditional SDFTs are found in [1] and [2], the next section provides a brief summary of SDFT properties necessary to derive the mSDFT.

Digital Object Identifier 10.1109/MSP.2010.938088

SLIDING DFT

Let us consider DFT computations in the time window of length M sliding along the signal x_n, such that $x_n = 0$ for $n < 0$

$$X_n^k = \sum_{m=0}^{M-1} x_{q+m} W_M^{-km}, \qquad (1)$$

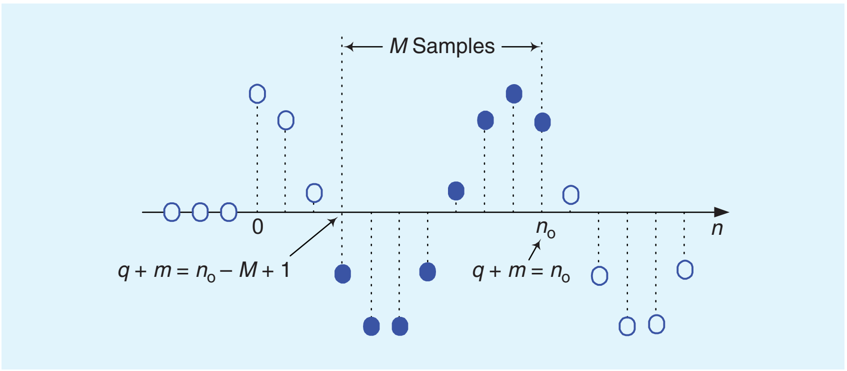
where $q = n - M + 1, 0 \le k \le M - 1$, and the complex twiddle factor equals $W_M = e^{j2\pi/M}$. To illustrate our time-domain indexing in (1), the solid dots in Figure 1 are the input x_n samples used to compute X_n^k when $n = n_o$.

In our notation, the superscript k in X_n^k is not a traditional exponent but instead refers to the DFT's frequency (kth-bin) index, and the subscript n is a time index. As such, k is a constant and the n in X_n^k indicates that the DFT is computed from samples $x_{n-M+1}, x_{n-M+2}, \ldots, x_n$. In the case of sample-by-sample signal processing, consecutive kth-bin DFT output samples X_n^k, X_{n+1}^k, $X_{n+2}^k, \ldots$, are computed from the x_n time samples that differ only by the first and last samples. A recursive formula for computing (1) may be derived as follows:

$$\begin{aligned} X_n^k &= \sum_{m=0}^{M-1} x_{q+m} W_M^{-km} \\ &= \sum_{m=0}^{M-1} x_{q+m-1} W_M^{-k(m-1)} - x_{q-1} W_M^{k} \\ &\quad + x_{q+M-1} W_M^{-k(M-1)} \\ &= W_M^k \sum_{m=0}^{M-1} x_{q+m-1} W_M^{-km} - x_{q-1} W_M^k \\ &\quad + x_{q+M-1} W_M^k \\ &= W_M^k (X_{n-1}^k - x_{q-1} + x_{q+M-1}) \\ &= W_M^k (X_{n-1}^k - x_{n-M} + x_n). \qquad (2) \end{aligned}$$

The structure of the difference in equation (2) is depicted in Figure 2(a). This traditional SDFT filter is only marginally stable because it has a z-domain pole located at $z = W_M^k$ on the unit circle as shown in Figure 2(b).

In practice, finite precision representation of the twiddle factor is the cause of instabilities and accumulated errors in the traditional SDFT. Except for when a pole is located at $z = \pm 1$ or $z = \pm j$, our nonideal numerical precision of the W_M^k coefficient places that pole either slightly inside, or slightly outside, the unit circle. When the pole is just inside the unit circle, a small error is induced in an X_n^k output sample. Those errors accumulate with each new X_n^k output sample computation. On the other hand, if the W_M^k pole is just outside the unit circle, the network in Figure 2(a) becomes unstable.



[FIG1] Solid dots represent the x_n samples used to compute X_n^k for $M = 8$.

It is easy to observe that for DFT bin $k = 0$ we have, from (2),

$$X_n^0 = X_{n-1}^0 - x_{n-M} + x_n. \quad (3)$$

Computation of successive values of X_n^0 requires simple summations of the input samples in the sliding time window of length M.

Due to the absence of the typically imprecise W_M^k coefficient, the recurrence in (3) is unconditionally stable and does not accumulate errors. In the next section, we use the DFT modulation property to effectively shift the DFT bin of interest to the position of $k = 0$ and then use (3) for computing that DFT bin output.

MODULATED SDFT

By the Fourier modulation property, the X^k DFT bin may be shifted to the index $k = 0$ (zero Hz) by the multiplication of the input signal x_n by the modulation sequence W_M^{-km} [4]

$$X_n^0 = X_{n-1}^0 - x_{n-M}W_M^{-k(m-M)} + x_n W_M^{-km}. \quad (4)$$

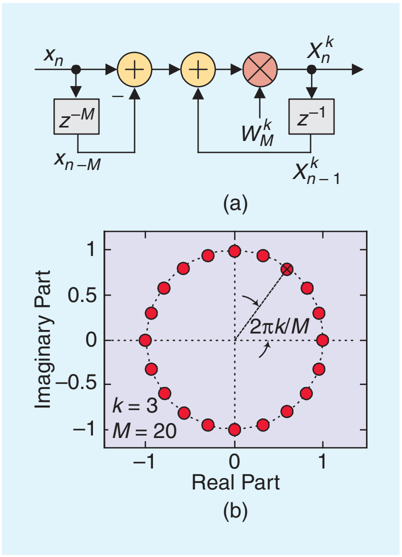
The W_M^{-km} term in (4) is the same complex twiddle factor used in the DFT definition in (1).

The structure of the mSDFT in (4) is depicted in Figure 3(a). Because there is no complex twiddle factor in the feedback of the resonator, the pole of the mSDFT resonator is located exactly (with no finite-precision numerical error) at $z = 1$ on the unit circle.

Unfortunately, if multiple DFT frequency bins are to be computed, which, for example, is the case for applying the time window in the frequency domain [1], one length-M delay buffer is needed in Figure 3(a) for each frequency bin. Note however, due to the periodicity of W_M^{-km}, the difference (4) may be rewritten in the desired form of

$$X_n^0 = X_{n-1}^0 + W_M^{-km}(-x_{n-M} + x_n). \quad (5)$$

The structure of our mSDFT in (5) is depicted in Figure 3(b). In the case of computing multiple DFT bins using (5),



[FIG2] Traditional SDFT: (a) structure and (b) *z*-plane pole/zero locations for *k* = 3 and *M* = 20.

memory is saved because only one delay buffer is required.

MODULATING SEQUENCE

The phase of the modulating sequence is changing with index m. For m we have W_M^{-km}, but for the next sample $m + 1$ we have

$$W_M^{-k(m+1)} = W_M^{-km} W_M^{-k}. \quad (6)$$

Thus, the phase of the modulating sequence equals 0 for $m = 0$ and increases by the W_M^{-k} in each iteration. From the definition in (1), it is seen that for $n = M$ the computations are started from $q = 1$, thus, the phase of the analyzed signal refers to the time instant $q = 1$. On the other hand, the phase of the modulating sequence for $q = 1$ equals W_M^{-k} and not 0; thus, we have a W_M^{-k} phase difference between the modulating sequence and the analyzed signal. Considering the above, we have following relation between the desired X_n^k, DFT of x_n, and the computed X_n^0 (4)

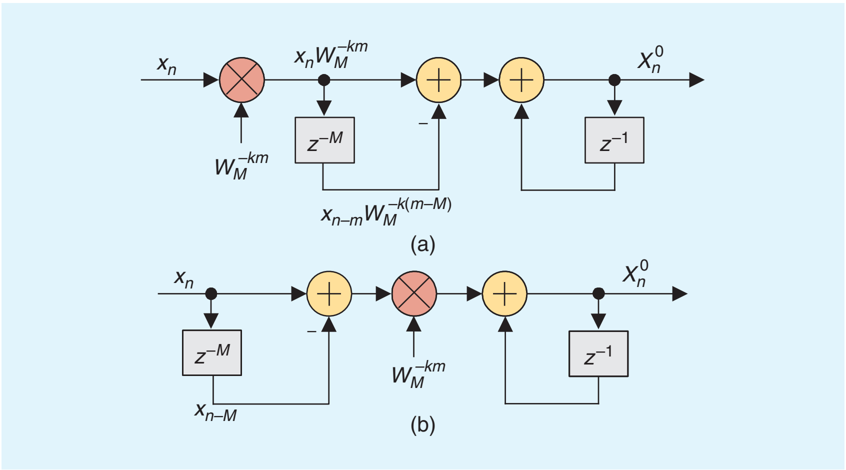
$$X_n^k = W_M^{k(m+1)} X_n^0. \quad (7)$$

The twiddle factor in (7) corrects the phase of X_n^0. It is seen from (7) that the modulus of X_n^k and X_n^0 is always the same, and the phase is the same when $W_M^{km} = 1$.

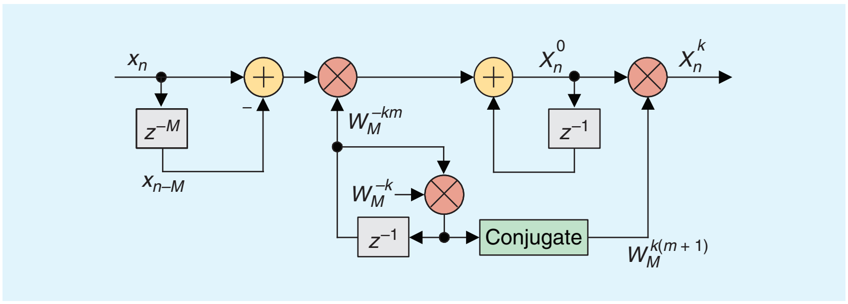
The drawback of the structure from Figure 3(b) as compared to Figure 2(a) is that the modulating sequence is time varying (it depends on the index m). That is, new sine and cosine factors must be computed, or stored in memory, for each new input x_n sample. To avoid that computational problem, in the following, we derive a simple form of the complex oscillator that will replace the modulating sequence in Figure 3(b). By shifting time indices in (6) we obtain the recurrence for modulating sequence

$$W_M^{-km} = W_M^{-k(m-1)} W_M^{-k}, \quad m = 0, 1, \ldots, M-1. \quad (8)$$

Note, that W_M^{-km} always has an integer number of periods in M samples as m cycles through the range $0 \le m \le M - 1$. Thus, W_M^{-km} is automatically restarted every M samples with the value



[FIG3] Modulated SDFT structures: (a) mSDFT in (4) and (b) mSDFT in (5).



[FIG4] The mSDFT structure with recursive computation of the modulating sequence in (8) and phase correction in (7).

$W_M^{-km} = 1$ when $m = 0$. This periodic restarting of W_M^{-km} prevents our computations from accumulated errors. The mSDFT with recursive computation of the modulating sequence in (8) is depicted in Figure 4.

The phase of the frequency bin X_n^0 computed by mSDFT is related to the phase of the desired DFT bin X_n^k by (7). Figure 4 depicts mSDFT with the phase correction. The output of the mSDFT in Figure 4 is the same as the output of the SDFT in Figure 2(a).

It is worth noting that if our spectrum analysis application requires only DFT magnitude results, then the final complex multiplication in Figure 4 is not needed because $|X_n^k| = |X_n^0|$.

THE mSDFT IS AN ATTRACTIVE OPTION FOR RECURSIVE COMPUTATION OF DFT BINS AND IS PREFERABLE TO OTHER SDFT ALGORITHMS.

OTHER STABLE SDFT ALGORITHMS

In this section, for the purpose of comparison with the mSDFT, we describe two other stable SDFT algorithms.

The SDFT recurrence (2) would be stable if we had used rW_M^k, with $0 \ll r < 1$, as the multiplier instead of W_M^k as was done in [2]. Let us define the following DFT:

$$\hat{X}_n^k = \mathrm{DFT}\{x_{q+m} r^{M-m}\} = \sum_{m=0}^{M-1} x_{q+m} r^{M-m} W_M^{-km}, \qquad (9)$$

where $q = n - M + 1, 0 \le k \le M - 1$, and $0 \ll r < 1$. A hat symbol in (9) and further equations denotes an approximation of the DFT as defined by (1). From (9) we derive, similar to (2), the recurrence

$$\hat{X}_n^k = rW_M^k(\hat{X}_{n-1}^k - x_{n-M} r^M + x_n). \quad (10)$$

From (9) it is seen that we compute the DFT of the signal multiplied by the sequence r^{M-m} and not the signal itself; thus, we trade accuracy for stability. We will refer to (10) as the rSDFT algorithm.

Another stable SDFT was proposed in [3]. In the following derivation of that stable SDFT, we assume, without the loss of generality, $M = 4$. Let us define the following DFT:

$$\begin{aligned}\hat{X}_{n+0}^k &= x_{q+0}W_4^{-k0} + x_{q+1}W_4^{-k1} \\ &\quad + x_{q+2}W_4^{-k2} + x_{q+3}W_4^{-k3} \\ \hat{X}_{n+1}^k &= x_{q+1}rW_4^{-k0} + x_{q+2}rW_4^{-k1} \\ &\quad + x_{q+3}rW_4^{-k2} + x_{q+4}W_4^{-k3} \\ \hat{X}_{n+2}^k &= x_{q+2}rW_4^{-k0} + x_{q+3}rW_4^{-k1} \\ &\quad + x_{q+4}W_4^{-k2} + x_{q+5}W_4^{-k3} \\ \hat{X}_{n+3}^k &= x_{q+3}rW_4^{-k0} + x_{q+4}W_4^{-k1} \\ &\quad + x_{q+5}W_4^{-k2} + x_{q+6}W_4^{-k3}.\end{aligned} \qquad (11)$$

In (11) there is no r in the upper equation; in the second equation we have r for $m = 0, 1, 2$; we have r for $m = 0, 1$ in the third equation; and one r for $m = 0$ in the last equation. The process of putting the r variable in the DFT definition repeats circularly. The definition in (11) may be computed recursively based on two difference equations [3]. From the first two lines of (11) and from the second and the third line of (11), we have the recurrence as shown in (12) at the bottom of the page.

We will refer to (12) as the Douglas and Soh (D&S) algorithm. It is seen from (10) and (12) that for $r = 1$, both algorithms become the SDFT in (2). In the next section we illustrate, by the example, the advantages of using the mSDFT rather than the rSDFT (10) or D&S (12) algorithms.

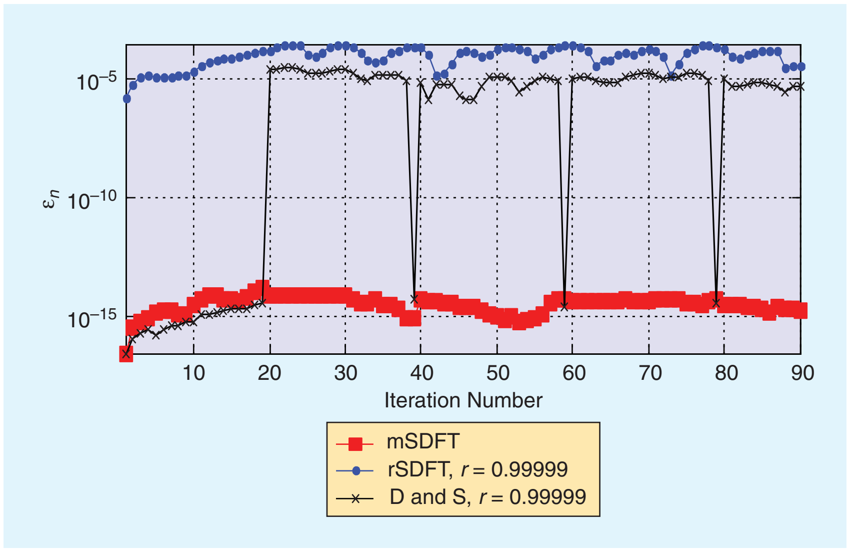
NUMERICAL SIMULATIONS

Figure 5 depicts the error of recursive DFT computations defined as

$$\varepsilon_n = |X_{n\,\mathrm{DFT}}^k - X_{n\,\mathrm{SDFT}}^k|, \qquad (13)$$

where $X_{n\,\mathrm{DFT}}^k$ denotes the standard DFT bin result computed from the batch of data of length M, and $X_{n\,\mathrm{SDFT}}^k$ stands for recursive computations of the mSDFT, rSDFT, and D&S algorithms. The test signal was zero-mean Gaussian noise with a standard deviation equal to one, the length of the DFT was $M = 20$, and the $k = 3$ bin was computed. The simulation was carried out in a 64-b double precision MATLAB environment. We see from Figure 5 that, in the given example, the mSDFT is approximately ten orders of magnitude more accurate than the rSDFT and D&S algorithms. The accuracy improvement depends on the value of$r$; for example, it is approximately seven orders of magnitude for $r = 0.999999999$ and 15 orders for $r = 0.9$. The accuracy of the mSDFT, as presented in Figure 5, may be degraded by the limited precision of the twiddle factor representation in Figure 4. But for equal precision twiddle factors, we observe better accuracy for the mSDFT than for the other two stable algorithms.

$$\hat{X}_n^k = \begin{cases} rW_M^k(\hat{X}_{n-1}^k - x_{n-M}) + W_M^k x_n, & \text{if } n \text{ modulo } M = 0 \\ W_M^k(\hat{X}_{n-1}^k - x_{n-M} r + x_n), & \text{otherwise.} \end{cases} \qquad (12)$$



[FIG5] **Error of single-bin DFT recursive computations for $M = 20$ and $k = 3$.**

Finally, we make some remarks about preferable arithmetic formats. As an example, consider $M = 4$, from (5) we have $X_5^0 = [W_4^{-k}x_1 + W_4^{-2k}x_2 + W_4^{-3k}x_3 + x_4 + W_4^{-k}x_5] - W_4^{-k}x_1$, where the values in brackets were previously summed. The mSDFT will run correctly for an unlimited time, without being reset, only if $X_5^0 = W_4^{-2k}x_2 + W_4^{-3k}x_3 + x_4 + W_4^{-k}x_5$ (and similarly for the next iterations) and this may be obtained only in integer arithmetic. Thus, fixed-point arithmetic is the best choice for implementing mSDFT.

In the case of floating-point arithmetic, we typically have $X_5^0 = W_4^{-2k}x_2 + W_4^{-3k}x_3 + x_4 + W_4^{-k}x_5 \pm \text{err}$, where err is a small roundoff error that may add or subtract. Let us assume that err is on the level of 10^{-10} and the worst case scenario that err only adds up, then it will take 10^{10} iterations to build up the overall error to a value of one. For example, at a sampling frequency of 44 kHz it will take at least 63 h of continual operation for the error to reach one. Thus, in practice, implementation in floating-point arithmetic may be sufficiently accurate for a sufficiently long time, although it may not run correctly for an unlimited time.

COMPUTATIONAL WORKLOAD

Table 1 gives a computational workload comparison of the various sliding DFT algorithms, with the assumption that the input signal is real and one complex multiplication requires four real multiplications and two additions, although it is also possible to compute one complex multiplication via three real multiplications and five additions [5].

[TABLE 1] COMPUTATION WORKLOAD PER OUTPUT SAMPLE.

METHOD	REAL MULTIPLIES	REAL ADDS
SDFT (2) [FIGURE 2(a)]	4	4
RSDFT ALGORITHM, (10)	5	4
D & S ALGORITHM, (12)	4, IF N MODULO M = 0, 5 OTHERWISE	5, IF N MODULO M = 0, 4 OTHERWISE
MSDFT TO COMPUTE X_n^0 (FIGURE 4)	6	5
MSDFT TO COMPUTE X_n^k (FIGURE 4)	10	7

SUMMARY

This article presented a novel method of computing the SDFT that we call the mSDFT. The accumulated errors and potential instabilities inherent in traditional SDFT algorithms are drastically reduced in the mSDFT. We removed the twiddle factor from the feedback in a traditional SDFT resonator and thus the finite precision of its representation is no longer a problem.

We compared other algorithms for ensuring stability of the SDFT described in [1]–[3] to the stable mSDFT. Our comparison showed those algorithms trade accuracy for stability, whereas the mSDFT is always stable and more accurate than other stable algorithms in the same computing environment. As such, the mSDFT is an attractive option for recursive computation of DFT bins and is preferable to other SDFT algorithms.

A MATLAB program that implements our mSDFT in Figure 4, the rSDFT in (10), and the D&S in (12) algorithms is available for download at: http://www.signalprocessingsociety.org/publications/periodicals/spm/columns-resources/#tips.

AUTHOR

Krzysztof Duda (kduda@agh.edu.pl) is an assistant professor in the Department of Measurement and Instrumentation at the AGH University of Science and Technology, Kraków, Poland.

REFERENCES

[1] E. Jacobsen and R. Lyons, "The sliding DFT," *IEEE Signal Processing Mag.*, vol. 20, no. 2, pp. 74–80, Mar. 2003.

[2] E. Jacobsen and R. Lyons, "An update to the sliding DFT," *IEEE Signal Processing Mag.*, vol. 21, no. 1, pp. 110–111, Jan. 2004.

[3] S. Douglas and J. Soh, "A numerically stable sliding-window estimator and its application to adaptive filters," in *Proc. 31st Annu. Asilomar Conf. Signals, Systems, and Computers*, vol. 1, Pacific Grove, CA, Nov. 1997, vol. 1, pp. 111–115.

[4] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, 2nd ed. Englewood-Cliffs, NJ: Prentice-Hall, 1999, p. 576.

[5] R. G. Lyons, *Understanding Digital Signal Processing*, 2nd ed. Englewood-Cliffs, NJ: Prentice-Hall, 2004, p. 487.

[SP]