Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

*Digital Object Identifier 10.1109/ACCESS.2017.Doi Number*

# Fast Hopping Discrete Sine Transform

**VITALY KOBER, (Member, IEEE)**
Center of Scientific Research and Higher Education of Ensenada, Ensenada, BC 22860, Mexico

Corresponding author: Vitaly Kober (e-mail: vkober@hotmail.com).

**ABSTRACT** Discrete sine transform (DST) is widely used in digital signal processing such as image coding, spectral analysis, feature extraction, and filtering. This is because the discrete sine transform is close to the optimal Karhunen–Loeve transform for first-order Markov stationary signals with low correlation coefficients. Short-time (hopping) discrete sine transform can be employed for time-frequency analysis and adaptive processing quasi-stationary data such as speech, biomedical, radar and communication signals. Hopping transform refers to a transform computed on the signal of a fixed-size window that slides over the signal with an integer hop step. In this paper, we first derive a second-order recursive equation between DST spectra in equidistant signal windows, and then propose two fast algorithms for computing the hopping DST based on the recursive relationship and input-pruned DST algorithm. The performance of the proposed algorithms with respect to computational costs and execution time is compared with that of conventional sliding and fast DST algorithms. The computational complexity of the developed algorithms is lower than any of the existing algorithms, resulting in significant time savings.

**INDEX TERMS** Discrete sine transform, hopping algorithm, short-time transform, sliding algorithm, signal processing

## I. INTRODUCTION

Discrete sine transform (DST) was first introduced for the processing of long-term stationary data [1], and later various versions of this transform were proposed [2], [3]. Unlike the Karhunen–Loeve transform, the DST is data independent and possesses a fast algorithm. The DST has found widespread use in digital signal processing such as data compression [4], adaptive digital filtering [5], image restoration [6], and interpolation [7], [8]. The performance of the DST is comparable to that of the discrete cosine transform (DCT) and, therefore, can be seen as a good alternative to the DCT. For signals with the correlation coefficient close to one, the DCT yields much better results than the DST. On the other hand, the DST performs better when the correlation coefficient is in the interval (-0.5, 0.5) [2].

Signal processing in the short-time domain [9] is a suitable technique for carrying out time-frequency analysis and processing of quasi-stationary signals. It can be applied to ECG signal processing [10], spectral analysis and speech processing [11], adaptive digital filtering [12], [13], radar emitter recognition [14], spectral analysis of biological signals [15], heart sound classification [16], time-frequency analysis of high-rate dynamic systems [17], etc. Short-time processing in the orthogonal transform domain can be realized by processing the signal in a window moving along the signal with an integer step. To obtain a reasonable spectral resolution, the window size must be large enough and, at the same time, the window size must be small enough so that the signal processed in the window is approximately stationary. In this case, short-time (hopping) transform [18] is a time series of equidistant windowed signal transforms. Recently, fast algorithms have been proposed to compute Hartley [19] and DCT [20] short-time transforms. The DST is an important orthogonal transform for time sequence analysis and may serve as an appropriate hopping transform for processing time-varying signals. Computing the DST in a window moving with one sample step on the signal is computationally expensive, so fast algorithms were proposed to compute four types of the transform using recursive equations [21], [22].

In this paper, two fast hopping DST algorithms with an arbitrary hop step are proposed. The algorithms can adjust the time hop between successive DST outputs. The work has the following contributions:

- Exploiting the z-transform technique, a recursive second-order equation is obtained for computing the hopping DST.

- Two fast hopping DST algorithms are proposed by employing the recursive equation and input-pruned DST algorithm.
- Computational cost and running time of the proposed algorithms are compared with those of known fast and sliding DST algorithms. The computational complexity of the proposed algorithms is the lowest among existing fast and sliding DST algorithms.

Organization of the paper is as follows. Section II introduces the notation and derives the relationship between three adjacent equidistant DST spectra. Two fast hopping DST algorithms are proposed in Section III. These algorithms are analyzed and discussed in Section IV. We conclude in Section V.

## II. RECURSIVE COMPUTATION OF HOPPING DST

Let us recall the definition of DST. We use the following notation: $\mathrm{cs}_N(rs) \equiv \cos\left(\frac{\pi}{N} rs\right)$, $\mathrm{sn}_N(rs) \equiv \sin\left(\frac{\pi}{N} rs\right)$, where $r$ is an integer, $N$ is the transform order, and $s = 1,\dots N-1$. Since the normalization factor $\sqrt{2/N}$ can be taken into account in the inverse transform, it is discarded.

Hopping DST with a hop step $p$ is defined as

$$y_s(kp) = \sum_{n=-N_1}^{N_2} x(kp+n)\,\mathrm{sn}_N\left(s(n+N_1+1)\right), \quad (1)$$

where $\{x(kp); k = ..,-N_1,-N_1+1,\dots 0,\ 1,\dots N_2,\ N_2+1..\}$ is the input signal; $\{y_s(kp),\ s=1,\dots N-1\}$ is the discrete sine transform at time $kp$; $N_1$ and $N_2$ are integers; $N = N_1 + N_2 + 2$ is the window size.

Three consecutive DST spectra are related as follows [22]:

$$y_s(k+2) - 2\,\mathrm{cs}_N(s)\, y_s(k+1) + y_s(k) = \delta_s(k)\,\mathrm{sn}_N(s), \quad (2)$$

where $\delta_s(k) = x(k-N_1) + (-1)^{s+1} x(k+N_2+2)$.

This is a linear difference inhomogeneous equation that converts into a linear difference equation with constant coefficients for fixed $s$. A linear casual time-invariant system defined by such an equation can be analyzed with the unilateral z-transform [9]. By applying the z-transform and then using its shift property, the following expression is obtained:

$$\begin{aligned} z^2\left[Y_s(z) - y_s(0) - z^{-1} y_s(1)\right] - 2\,\mathrm{cs}_N(s)\, z\left[Y_s(z) - y_s(0)\right] \\ + Y_s(z) = D_s(z)\,\mathrm{sn}_N(s) \end{aligned}, \quad (3)$$

where $Y_s(z)$ and $D_s(z)$ are the z-transforms of $y_s(k)$ and $\delta_s(k)$, respectively. $Y_s(z)$ is expressed as

$$Y_s(z) = \left[D_s(z)\,\mathrm{sn}_N(s)\, z^{-1} + y_s(0)\left[z - 2\,\mathrm{cs}_N(s)\right] + y_s(1)\right] T_s(z), \quad (4)$$

with

$$T_s(z) = \frac{z^{-1}}{1 - 2\,\mathrm{cs}_N(s)\, z^{-1} + z^{-2}}.$$

$T_s(z)$ can be represented as follows:

$$T_s(z) = \frac{1}{q_1(s) - q_2(s)}\left(\frac{1}{1 - q_1(s)_1 z^{-1}} - \frac{1}{1 - q_2(s) z^{-1}}\right), \quad (5)$$

where $q_1(s) = \exp\left(j\frac{\pi}{N} s\right)$ and $q_2(s) = \exp\left(-j\frac{\pi}{N} s\right)$ are the roots of the denominator of $T_s(z)$.

The inverse transform of $T_s(z)$ can be computed as

$$t_s(k) = \frac{q_1^k(s) - q_2^k(s)}{q_1(s) - q_2(s)} u(k-1) = \frac{\mathrm{sn}_N(ks)}{\mathrm{sn}_N(s)} u(k-1), \quad (6)$$

where $u(k)$ is defined as 1, for $k \ge 0$, and 0, for $k < 0$.

Using the convolution and shift properties of the z-transform, the inverse transform of (4) is given as

$$\begin{aligned} y_s(k) = \mathrm{sn}_N(s) \sum_{r=0}^{k} t_s(r-1)\delta_s(k-r) + y_s(1) t_s(k) \\ + y_s(0)\left[t_s(k+1) - 2\,\mathrm{cs}_N(s) t_s(k)\right] \end{aligned}. \quad (7)$$

Taking into account that $t_s(k) = 0$ for $k \le 1$, $q_1(s) q_2(s) = 1$, $q_1(s) + q_2(s) = 2\,\mathrm{cs}_N(s)$ and substituting (6) into (7), for $k \ge 2$ we get

$$\begin{aligned} y_s(k) = \sum_{r=1}^{k-1} \mathrm{sn}_N(rs)\,\delta_s(k-r-1) \\ - \frac{y_s(0)\,\mathrm{sn}_N((k-1)s) - y_s(1)\,\mathrm{sn}_N(ks)}{\mathrm{sn}_N(s)} \end{aligned}. \quad (8)$$

Suppose that $y_s(0)$ and $y_s(p)$ $(p > 1)$ are given, we obtain $y_s(1)$ from (8) as follows:

$$\begin{aligned} y_s(1) = \frac{y_s(p)\,\mathrm{sn}_N(s) + y(0)\,\mathrm{sn}_N((p-1)s)}{\mathrm{sn}_N(ps)} \\ - \sum_{r=1}^{p-1} \delta_s(p-r-1)\,\mathrm{sn}_N(rs) \end{aligned}. \quad (9)$$

Substituting (9) into (8), we express $y_s(k)$ for any integer $k$ as

$$\begin{aligned} y_s(k) = \frac{\mathrm{sn}_N((k-p)s)}{\mathrm{sn}_N(ps)}\left(\sum_{r=1}^{p-1} \delta_s(r-1)\,\mathrm{sn}_N(rs) - y_s(0)\right) \\ + \frac{\mathrm{sn}_N(ks)}{\mathrm{sn}_N(ps)} y_s(p) + \sum_{r=p}^{k-1} \delta_s(r-1)\,\mathrm{sn}_N((k-r)s) \end{aligned}. \quad (10)$$

Finally, the relationship between three adjacent equidistant DST spectra at times $2p$, $p$, and 0 is written as

$$\begin{aligned} y_s(2p) = \sum_{r=1}^{p-1} \left(\delta_s(r-1) + \delta_s(2p-r-1)\right)\mathrm{sn}_N(rs) - y_s(0) \\ + \delta_s(p-1)\,\mathrm{sn}_N(ps) + 2\,\mathrm{cs}_N(ps)\, y_s(p) \end{aligned}. \quad (11)$$

## III. FAST HOPPING DST ALGORITHMS

Let us rewrite (11) for $p>1$ as follows:

$$y_s(2p)=\sum_{r=1}^{p}A_s(r)\mathrm{sn}_N(rs)-y_s(0)+2\,\mathrm{cs}_N(ps)\,y_s(p),\quad(12)$$

where $\{A_s(r)=\delta_s(r-1)+\delta_s(2p-r-1);\ r=1,\dots p-1\}$, and $A_s(p)=\delta_s(p-1)$. The number of additions per window required for calculating $\{A_s(r);\ r=1,\dots p\}$ is $4p-2$. Note that the calcultion of $\delta_s(r)$ requires two additions (for even and odd $s$), and the coefficients $\{\delta_s(r);\ r=0,\dots p-1\}$ have already been calculated and stored at time $p$.

### A. FAST ALGORITHM BASED ON PROPERTIES OF DISCRETE SINUSOIDAL FUNCTIONS

Let us analyze the symmetry property of $\{\mathrm{sn}_N(rs);\ s=1,\dots N-1,\ r=1,\dots p\}$. Suppose that $g_r$ is the greatest common factor of $r$ and $N$. Table 1 shows the special values of the discrete functions; here $N$, $r$ and $l$ are arbitrary integers.

TABLE I
SPECIAL VALUES OF SINUSOIDAL FUNCTIONS

| FUNCTIONS | VALUES | | |
|---|---|---|---|
| | 0 | 1 | -1 |
| $sn_N(rs)$ | $s=\frac{N}{r}l$ | $s=\frac{N}{2}\frac{(1+4l)}{r}$ | $s=\frac{N}{2}\frac{(-1+4l)}{r}$ |
| $cs_N(rs)$ | $s=\frac{N}{2r}(2l+1)$ | $s=\frac{2N}{r}l$ | $s=\frac{N}{r}(2l+1)$ |

For fixed $r$, the number of ones and zeros of the function is equal to $g_r$ and $g_r-1$, respectively (first line of Table I). The quantity of zeros of $\{cs_N(ps);\ s=1,\dots N-1\}$ is equal to $g_p$ (second line of Table I). Let us calculate the integers: $\bar{r}_r=r/g_r$ and $\bar{N}_r=N/g_r$. The discrete function $\left\{\left|\mathrm{sn}_{\bar{N}_r}(rs)\right|=\left|\mathrm{sn}_{\bar{N}_r}\left(r\left(\bar{N}_r-s\right)\right)\right|;\ s=1,\dots\bar{N}_r-1,\quad r=1,\dots\bar{r}_r\right\}$ has symmetry about the point $\left(\bar{N}_r-1\right)/2$. In addition, for $g_r>1$ the function is periodic with a period of $\bar{N}_r$; that is, $\left\{\left|\mathrm{sn}_{\bar{N}_r}(rs)\right|=\left|\mathrm{sn}_{\bar{N}_r}\left(r\left(s+l\bar{N}_r\right)\right)\right|;\quad s=1,\dots\bar{N}_r-1,\ l=1,\dots g_r-1\right.$, $\left.r=1,\dots\bar{r}_r\right\}$. Assume that $N$ is even and $r$ is fixed. For even $\bar{N}_r$, from periodicity and symmetry of the functions: $\left\{\left|\mathrm{sn}_{\bar{N}_r}(rs)\right|=\left|\mathrm{sn}_{\bar{N}_r}\left(r\left(l\bar{N}_r\pm s\right)\right)\right|,\ (-1)^s=(-1)^{l\bar{N}_r\pm s}\right.$,

$\left.A_s(r)=A_{l\bar{N}_r\pm s}(r);\ l=1,\dots g_r-1,\ r=1,\dots\bar{r},\quad s=1,\dots\bar{N}_r-1\right\}$,

one can estimate the number of multiplications required to calculate the first term of (12) as $C^r_{MUL}=\left[\frac{\bar{N}_r-1}{2}\right]$. Here $[x/y]$ is the integer quotient. For odd $\bar{N}_r$, from periodicity and symmetry of the functions: $\left\{\left|\mathrm{sn}_{\bar{N}_r}(rs)\right|=\left|\mathrm{sn}_{\bar{N}_r}\left(r\left(2l\bar{N}_r\pm s\right)\right)\right|,\ (-1)^s=(-1)^{2l\bar{N}_r\pm s}\right.$,

$\left.A_s(r)=A_{2l\bar{N}_r\pm s}(r);\ l=1,\dots g_r-1,\quad r=1,\dots\bar{r},\ s=1,\dots\bar{N}_r-1\right\}$,

the quantity of multiplications is estimated as $C^r_{MUL}=\bar{N}_r-1$. For odd $N$ and fixed $r$, the number of multiplications is $C^r_{MUL}=\bar{N}_r-1$. Thus, the total number of multiplications per window for computing the hopping DST can be estimated as

$$C_{MUL}=(N-1)+\sum_{r=1}^{p}C^r_{MUL}-g_p.\qquad(13)$$

The total number of additions per window is equals to

$$C_{ADD}=(N-1)(p+1)+(4p-2)-\sum_{r=1}^{p}(g_r-1)-g_p.\quad(14)$$

Additional costs are required to calculate the initial $p$ coefficients. We call this algorithm ALG-1. Note that the window size for the proposed algorithm is any integer determined by the characteristics of the processed signal.

Next, we give a simple example for computing the hopping DST coefficients for $p=2$, $N_1=7$, $N_2=7$ and $N=16$. In other words, the output DST coefficients $\{y_s(2p),\ s=1,\dots15\}$ are computed at time $2p$. We borrow two coefficients $\Delta_1^+=x_{-7}+x_9;\ \Delta_1^-=x_{-7}-x_9$ from time $p$ and pre-calculate the auxiliary data:

$\Delta_2^+=x_{-6}+x_{10};\ \Delta_2^-=x_{-6}-x_{10};\ \Delta_3^+=x_{-5}+x_{11};\ \Delta_3^-=x_{-6}-x_{11}$

$A^+=\Delta_1^++\Delta_3^+;\ A^-=\Delta_1^--\Delta_3^-$

$S_1=0.1951A^+;\ S_2=0.3827A^-;\ S_3=0.5556A^+;\ S_4=0.7071A^-$

$S_5=0.8315A^+;\ S_6=0.9239A^-;\ S_3=0.9808A^+$

$\mathrm{Q}_1=0.3827\Delta_2^+;\ \mathrm{Q}_2=0.7071\Delta_2^-;\ \mathrm{Q}_3=0.9239\Delta_2^+$

The DST coefficients are calculated as follows:

$y_1(2p)=S_1-y_1(0)+1.8478y_1(p)+Q_1$

$y_2(2p)=S_2-y_2(0)+1.4142y_2(p)+Q_2$

$y_3(2p)=S_3-y_3(0)+0.7654y_3(p)+Q_3$

$y_4(2p)=S_4-y_4(0)+\Delta_2^-$

$y_5(2p)=S_5-y_5(0)-0.7654y_5(p)+Q_3$

$y_6(2p)=S_6-y_6(0)-1.4142y_6(p)+Q_2$

$y_7(2p)=S_7-y_7(0)-1.8478y_7(p)+Q_1$

$y_8(2p)=A^--y_8(0)-2y_8(p)$

$$y_9(2p) = S_7 - y_9(0) - 1.8478 y_9(p) - Q_1$$
$$y_{10}(2p) = S_6 - y_{10}(0) - 1.4142 y_{10}(p) - Q_2$$
$$y_{11}(2p) = S_5 - y_{11}(0) - 0.7654 y_{11}(p) - Q_3$$
$$y_{12}(2p) = S_4 - y_{12}(0) - \Delta_2^-$$
$$y_{13}(2p) = S_3 - y_{13}(0) + 0.7654 y_{13}(p) - Q_3$$
$$y_{14}(2p) = S_2 - y_{14}(0) + 1.4142 y_{14}(p) - Q_2$$
$$y_{15}(2p) = S_1 - y_{15}(0) + 1.8478 y_{15}(p) - Q_1.$$

One can observe that the algorithm complexity is 23 multiplications and 48 additions.

### B. FAST ALGORITHM BASED ON PRUNED DST

Let us define the first term of (12) as follows:

$$X_N(s) = \sum_{r=1}^{p} A_s(r)\,\mathrm{sn}_N(rs),\ s = 1,2,\ldots N-1. \quad (15)$$

It can be seen that only a subset of the input coefficients for $p < N-1$ is employed to compute the output DST. This method is referred to as input-pruned DST. Suppose that $\{A_s(r) = 0,\ r > p\}$, $p > 1$, and $N$ is of a power of 2, then the decimation-in-time radix-2 algorithm [23] recursively splits the DST into two half-length DSTs of the even-indexed and odd-indexed time samples as follows:

$$\begin{array}{l} X_N(s) = X^o_{N/2}(s) + X^e_{N/2}(s) \\ X_N(N-s) = X^o_{N/2}(s) - X^e_{N/2}(s) \end{array},\ s = 1,2,\ldots,N/2-1, \quad (16)$$

where

$$\begin{array}{l} X^o_{N/2}(s) = \dfrac{1}{2\,\mathrm{cs}_N(s)} \sum_{r=1}^{L_1} \left(A_s(2r-1) + A_s(2r+1)\right)\mathrm{sn}_{N/2}(rs) \\ X^e_{N/2}(s) = \sum_{r=1}^{L_2} A_s(2r)\,\mathrm{sn}_{N/2}(rs) \end{array}, \quad (17)$$

with $s = 1,2,\ldots,N/2-1$, and

$$X_{N/2}(N/2) = \sum_{r=0}^{L_1} A_{N/2}(2r+1)(-1)^r, \quad (18)$$

here $L_1 = [(p+1)/2] - 1$ and $L_2 = [p/2]$.

The decomposition is used recursively, and the transform size is halved each time. Note that the fast input-pruned DST algorithm is defined by a simple structured recursive matrix factorization of the transform matrix and represented by a regular signal flow graph. Fig. 1 shows the flow graph for $N = 16$ and $p = 7$. Solid lines with arrows represent unity transfer factors while dashed lines (red color) represent transfer factors of $-1$. A circle represents addition if there is more than one input line to the left. ↓ means multiplication by the corresponding factor $\left\{C_s = \dfrac{1}{2\,\mathrm{cs}_{16}(s)}, s = 1,2,\ldots 7\right\}$

For $p = 2^\mu - 1$ ($\mu$ is integer and $\mu > 1$), the complexity of the input-pruned DST algorithm in terms of additions and multiplications can be estimated as $DST_{ADD}(N,p) = N\dfrac{\mu(p+1)-2}{p+1} + (p+1)(\mu-4) + 6$ and $DST_{MUL}(N,p) = N\mu/2 - p$, respectively.

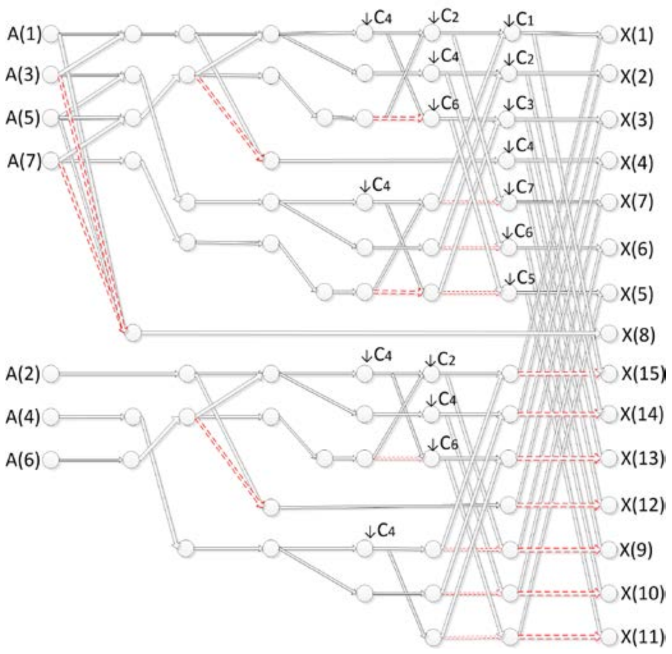

**FIGURE 1.** **Flow graph for the input-pruned DST computation, *N*=16 and *p*=7.**

For other values of $p$, the complexity can be estimated using the recursive equations (13)-(18). For example, for $p = 2$, the quantity of multiplications and additions is given as $DST_{MUL}(N,p) = 3N/4 - 2$ and $DST_{ADD}(N,p) = N - 2$, respectively. The total number of multiplications per window for computing the hopping DST is estimated as

$$C_{MUL} = (N-1) + DST_{MUL}(N,p) - g_p. \quad (19)$$

The total number of additions per window is equal to

$$C_{ADD} = 2(N-1) + (4p-2) + DST_{ADD}(N,p) - g_p. \quad (20)$$

Additional costs are required to calculate the initial $p$ coefficients. We refer to this algorithm as ALG-2.

A pseudo-code of the proposed HDFT algorithms is given in Table II. The algorithms require 2$N$ -2 memory locations to store the DST coefficients computed at times $p$ and 0. Since recursive computation of the output DST are carried out “in-place” using the memory originally occupied by the DST computed at time 0, no additional memory is s required to store the output data. Step 1 requires $(N-1)p + 6p - 2$ additional memory locations for storing

$\delta_s(k)$, $A_s(r)$ coefficients and trigonometric weighting factors. Step 2 requires $3N/2-2$ additional memory locations to store $X_N(s)$ and $C_s$ coefficients. Step 3 does not require additional memory.

TABLE II
PSEUDO-CODE OF THE PROPOSED HDST ALGORITHMS

**Input:** input samples $x(k)$ and DST outputs $y_s(p)$ and $y_s(0)$ at times $p$ and 0, respectively; $N = N_1 + N_2 + 2$ is the window size, $N_1$ and $N_2$ are integers; $p$ is the hop step.

**Output:** DST output $y_s(2p)$ at time 2$p$.

```
/* Step 1: setup */
for s=1 : N-1 do
    for r=1 : p-1 do
        $\mathrm{sn}_N(r,s) = \sin\left(\frac{\pi}{N} rs\right)$
    end for
    $\mathrm{cn}\,2_N(s) = 2\cos\left(\frac{\pi}{N} ps\right)$
end for
for s=1 : 2 do
    for k=0 : 2p-2 do
        $\delta(k,s) = x(k-N_1) + (-1)^{s+1} x(k+N_2+2)$
    end for
    for r=1 : p-1 do
        $A(r,s) = \delta(r-1,s) + \delta(2p-r-1,s)$
    end for
    $A(p,s) = \delta(p-1,s)$
end for

/*Step 2: input-pruned radix 2 DST (N is of a power of 2) */
for s=1 : $N/2-1$ do
    $C(s) = \left(2\cos\left(\frac{\pi}{N} s\right)\right)^{-1}$
end for
for k=1 : $\log_2 N$ do
    Compute $X_N(s)$ using equations (16), (17) and (18).
end for

/*Step 3: recursive computation */
for s=1 : N-1 do
    Compute $y_s(2p)$ using equation (12). For N is of a power of 2,
    the first term of (12) is replaced by $X_N(s)$.
end for

return $y_s(2p)$
```

## IV. SIMULATION RESULTS

In this section, we analyze the algorithms presented in the paper with respect to computational costs and execution time. The most popular among fast DST algorithms are fast radix-2 [24], [25]. The sliding DST algorithm [22] is executed $p$ times to calculate the equidistant DST spectra. For $p = 2$ and varying $N$, Tables III and IV show the performance in terms of multiplications and additions, respectively, the following tested algorithms: FDST is the fast algorithm [24], SDST is the sliding algorithm [22], ALG-1 and ALG-2 are the proposed algorithms.

TABLE III
PERFORMANCE OF ALGORITHMS IN TERMS OF MULTIPLICATIONS, P=2, N=$2^M$.

| ALGORITHMS | NUMBER OF OPERATIONS | N - LENGTH OF HOPPING WINDOW | | | | | |
|---|---|---|---|---|---|---|---|
| | | 16 | 32 | 64 | 128 | 256 | 512 |
| FDST | MN/2-N+1 | 17 | 49 | 129 | 321 | 769 | 1793 |
| SDST | (3N/2-2)P | 44 | 92 | 188 | 380 | 764 | 1532 |
| ALG-1 | EQ. (13) | 23 | 51 | 107 | 219 | 443 | 891 |
| ALG-2 | EQ. (19) | 23 | 51 | 107 | 219 | 443 | 891 |

TABLE IV
PERFORMANCE OF ALGORITHMS IN TERMS OF ADDITIONS, P=2, N=$2^M$.

| ALGORITHMS | NUMBER OF OPERATIONS | N - LENGTH OF HOPPING WINDOW | | | | | |
|---|---|---|---|---|---|---|---|
| | | 16 | 32 | 64 | 128 | 256 | 512 |
| FDST | 3MN/2-2N-M+2 | 62 | 173 | 444 | 1083 | 2554 | 5881 |
| SDST | (3N/2-2)P | 64 | 128 | 256 | 512 | 1024 | 2048 |
| ALG-1 | EQ. (13) | 48 | 96 | 192 | 384 | 768 | 1536 |
| ALG-2 | EQ. (19) | 48 | 96 | 192 | 384 | 768 | 1536 |

Note that the proposed algorithms have the same complexity, and for $N > 16$ outperform the fast and sliding DST algorithms. The execution times for floating point addition and multiplication in modern processors are comparable. Therefore, the algorithm complexity can be estimated by the number of flops (real multiplications and additions). Comparison of the tested algorithms in terms of flops for $N = 256$ and varying $p$ is given in Table V.

TABLE V
PERFORMANCE OF ALGORITHMS WITH RESPECT TO FLOPS, N=256.

| HOP STEP P | ALGORITHMS | | | |
|---|---|---|---|---|
| | FDST | SDFT | ALG-1 | ALG-2 |
| 2 | 3323 | 1788 | 1211 | 1211 |
| 3 | 3323 | 2682 | 1599 | 1439 |
| 4 | 3323 | 3576 | 1880 | 1563 |
| 5 | 3323 | 4470 | 2272 | 1610 |
| 6 | 3323 | 5364 | 2591 | 1736 |
| 7 | 3323 | 6258 | 2979 | 1868 |
| 15 | 3323 | 13410 | 5719 | 2316 |
| 31 | 3323 | 27714 | 11183 | 2796 |
| 63 | 3323 | 56322 | 22091 | 3380 |

One can observe that for $p > 1$ the proposed algorithms are faster than the sliding DST algorithm. The algorithm ALG-1 is more efficient than the fast DST algorithm when $p \le 7$. It can be seen that for $p < 63$ the algorithm ALG-2 is superior to the fast DST algorithm. The algorithm ALG-2 becomes faster than the algorithm ALG-1 when $p > 2$. As the window size increases, the boundary step values at which the fast DST algorithm is still no better than the proposed algorithms also increase.

We implemented all tested algorithms on a laptop with Intel Core i7-2630QM and 8 GB of RAM using MATLAB R2016a. To guarantee statistically correct results, we repeated all experiments 100 times and calculated the average runtime result for each algorithm. Fig. 2 shows the runtime performance of the algorithms.

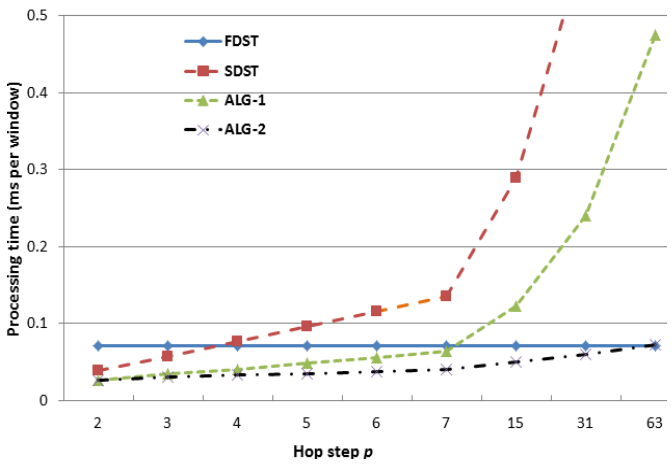

**FIGURE 2. Flow graph for the input-**pruned DST computation **Measured running time (milliseconds) of the algorithms per window for *N*= 256 and varying *p*.**

It can be noted that the theoretical results are in good accordance with the experimental results presented in Table V.

There are several types of DST [2], [3], which are suitable for processing different signal models. In this paper, fast hopping DST algorithms have been suggested for only one type of discrete sine transform (DST-I). The same approach can be used to design fast hopping algorithms for other types of DST, which will efficiently handle different signal models.

## V. CONCLUSION

Recursive equation between three adjacent equidistant DST spectra was obtained with the help of the z-transform. Using the recursive equation, input-pruned DST and properties of sinusoidal functions, two fast hopping DST algorithms have been proposed. The complexity of the hopping algorithms was compared with that of known fast and sliding DST algorithms. For the proposed algorithm ALG-1, the length of the hopping window can be arbitrary, determined by the characteristics of the processed signal. For the window length of a power of 2, the algorithm ALG-2 outperforms fast and sliding DST algorithms in a wide range of parameters. It was also shown that the obtained theoretical results are in good agreement with the presented experimental results.

## REFERENCES

[1] A. K. Jain, “A fast Karhunen-Loeve transform for a class of random processes,” *IEEE Trans. Commun*., vol. 24, no. 9, pp. 1023-1029, Sept. 1976. DOI: 10.1109/TCOM.1976.1093409.

[2] A. K. Jain, “A sinusoidal family of unitary transforms,” *IEEE Trans. Patt. Anal. Machine Intell*., vol. PAMI-I, pp. 356-365, Sept. 1979. DOI: 10.1109/TPAMI.1979.4766944.

[3] Z. Wang and B. Hunt, “The discrete W transform,” *Applied Math* Computat., vol. 16, pp. 19-48, Jan. 1985. DOI: 10.1016/0096-3003(85)90008-6.

[4] K. Rose, A. Heiman, and I. Dinstein, "DCT/DST alternate-transform image coding," *IEEE Trans. Commun*, vol. 38, no. 1, pp. 94-101, Jan. 1990. DOI: 10.1109/26.46533.

[5] J. Lee and C. Un, "Performance of transform-domain LMS adaptive digital filters," *IEEE Trans. Acoust., Speech, Signal Process*., vol. 34, no. 3, pp. 499-510, June 1986. DOI: 10.1109/TASSP.1986.1164850.

[6] S. Cheng, “Application of the sine transform method in time of flight positron emission image reconstruction algorithms,” *IEEE Trans. Biomed. Eng*., vol. BME-32, no. 3, pp. 185-192, March 1985. DOI: 10.1109/TBME.1985.325527.

[7] Z. Wang and L. Wang, “Interpolation using the fast discrete sine transform,” *Signal Process*., vol. 26, pp. 131-137, Jan. 1992. DOI: 10.1016/0165-1684(92)90059-6.

[8] M. Kim and Y.-L. Lee, “Discrete sine transform-based interpolation filter for video compression,” *Symmetry*, vol. 9, no. 11, p. 257, Nov. 2017. DOI: 10.3390/sym9110257.

[9] A. V. Oppenheim and R. W. Schafer, *Discrete-time signal processing*, 3rd ed., Upper Saddle River, NJ, USA: Prentice-Hall, 2009.

[10] R. R. Sharma, M. Kumar, and R. B. Pachori, “Joint time-frequency domain-based CAD disease sensing system using ECG signals," *IEEE Sensors J*., vol. 19, no. 10, pp. 3912-3920, May 2019, 10.1109/JSEN.2019.2894706.

[11] M. Portnoff, "Short-time Fourier analysis of sampled speech," *IEEE Trans. Acoust., Speech, Signal Process*., vol. 29, no. 3, pp. 364-373, June 1981. DOI: 10.1109/TASSP.1981.1163580.

[12] J. Shi, J. Zheng, X. Liu, W. Xiang, and Q. Zhang, "Novel short-time fractional Fourier transform: theory, implementation, and applications," *IEEE Trans. Signal Process*., vol. 68, pp. 3280-3295, May 2020. DOI: 10.1109/TSP.2020.2992865.

[13] V. Kober, “Robust and efficient algorithm of image enhancement,” *IEEE Trans. Consum. Electron*., vol. 52, no. 2, pp. 655–659, May 2006. DOI: 10.1109/TCE.2006.1649693.

[14] X. Wang, G. Huang, Z. Zhou, W. Tian, J. Yao, and J. Gao, “Radar emitter recognition based on the energy cumulant of short-time Fourier transform and reinforced deep belief network,” *Sensors*, vol. 18, bo. 9, p. 3103, August 2018. DOI: 10.3390/s18093103.

[15] Y. Wang, K. C. Veluvolu, “Time-frequency analysis of non-stationary biological signals with sparse linear regression based Fourier linear combiner,” *Sensors*, vol. 17, no. 6, p. 1386, Jun. 2017, doi:10.3390/s17061386.

[16] A. Thalmayer, S. Zeising, G. Fischer, J. Kirchner, “A robust and real-time capable envelope-based algorithm for heart sound classification: validation under different physiological conditions,” *Sensors*. Vol 20, no. 4, p. 972, Jun. 2020, doi: 10.3390/s20040972

[17] J. Yan, S. Laflamme, P. Singh, A. Sadhu, J.A. Dodson, “Comparison of time-frequency methods for real-time application to high-rate dynamic systems,” *Vibration*, vol. 3, no. 3, pp. 204-216, Aug. 2020, doi:10.3390/vibration3030016.

[18] C. Park and S. Ko, "The hopping discrete Fourier transform," *IEEE Signal Process. Mag*., vol. 31, no. 2, pp. 135-139, Mar. 2014. DOI: 10.1109/MSP.2013.2292891.

[19] V. Kober, “Fast recursive computation of sliding DHT with arbitrary step,” *Sensors*, vol. 20, no. 19, p. 5556, Sept. 2020, doi: 10.3390/s20195556.

[20] V. Kober, “Recursive algorithms for computing sliding DCT with arbitrary step,” *IEEE Sensors*, vol. 21, no. 10, pp. 11507-11513, May 2021, doi: 10.1109/JSEN.2020.3023892.

[21] V. Kober, "Fast recursive algorithm for sliding discrete sine transform", *Electron. Lett.*, vol. 38, no. 25, pp. 1747-1748, Dec. 2002. DOI: 10.1049/el:20021098.

[22] V. Kober, "Fast algorithms for the computation of sliding discrete sinusoidal transforms," *IEEE Trans. Signal Process.*, vol. 52, no. 6, pp. 1704–1710, Jun. 2004. DOI: 10.1109/TSP.2004.827184.

[23] P. Yip and K. R. Rao, "Fast decimation-in-time algorithms for a family of discrete sine and cosine transforms", *Circuits, Syst., Signal Process.*, vol. 3, pp. 387-408, Dec. 1984. DOI: 10.1007/BF01599167.

[24] Z. Wang, "Fast discrete sine transform algorithms," *Signal Process.*, vol. 19, no. 2, pp. 91-102, Feb. 1990. DOI:10.1016/0165-1684(90)90033-U.

[25] A. Gupta and K. R. Rao, "A fast recursive algorithm for the discrete sine transform", *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 38, no. 3, pp. 553–557, Mar. 1990. DOI: 10.1109/29.106875.