A Novel Discrete Transform

W. A. Gunarathna[1#] and H. M. Nasir[2]

[1]*Department of Mathematics, Faculty of Engineering, General Sir John Kotelawala Defence University, Sri Lanka*
[2]*Department of Mathematics and Statistics, College of Science*
Sultan Qaboos University, Oman

[#]gunarathnawa@yahoo.com

Abstract*—The notion of a discrete transform is of great importance in solving many problems in Science and Engineering. For instance, the ordinary discrete Fourier transform (DFT) is one of the most important and distinguished elements in the class of discrete transform which is extensively used in Digital Signal and Image processing. In this paper, we ourselves define a novel discrete transform. To define this, let* $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ *be a given sequence of* N *complex numbers. For a given positive integer* p *and a complex parameter* σ*, we define our transform by the sequence*

$$\alpha=\{\alpha_0(p),\alpha_1(p),\ldots,\alpha_{N-1}(p)\},$$

where

$$\alpha_k(p)=\sum_{j=0}^{N-1}\beta_j(\sigma+w^{jk})^p$$

for all $k=0,1,\ldots,N-1$ *and* $w=e^{-i2\pi/N}$ *is an* N^{th} *root of unity.*

We show that this transform holds the properties of the linearity and periodicity. The naive computation of this new transform requires a complexity of $O(pN^2)$ *which is computationally prohibitive for large values of* N *and* p*. For relatively small values of* p*, We further develop a fast algorithm with the complexity of* $O(N\log N)$*. The naive and fast algorithms are both implemented in MATLAB and we explore the performance of the fast algorithm by means of numerical examples.*

Keywords— Discrete Fourier Transform, Fast Fourier Transform, Novel Discrete Transform.

I. INTRODUCTION

Discrete transforms are widely used in many Science and Engineering applications, including Physics, Medical imaging, global weather forecasting, and Statistical analysis. For instance, one of the most important and distinguished elements in the class of discrete transforms is the ordinary discrete Fourier transform (DFT), where Signal and Image processing abound with many significant computations carried out by the DFT.

Let $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ be a sequence of N complex numbers. Then, its N -point DFT is defined by the sequence of N complex numbers $\{\alpha_0,\alpha_1,\ldots,\alpha_{N-1}\}$, where

$$\alpha_k=\sum_{j=0}^{N-1}\beta_j w^{jk}$$

for $k=0,1,\ldots,N-1$, where $w=e^{-i2\pi/N}$ is an N^{th} root of unity.

The naive computation of the DFT is prohibitive in terms of number of operations and computer times for large values of N. The fast Fourier transform (FFT), demonstrates outstanding performance in the digital computer, is a fast and robust algorithm for the DFT which was invented by Cooly and Tukey in 1965 and published in Cooley JW. & Tukey JW (1965). The FFT is so fast that it requires $3N/2\log N$ arithmetic operations, opposed to $O(N^2)$ arithmetic operations arising from the naive computation of the DFT. The discrete sine transform, discrete cosine transform, discrete polynomial transform, and spherical harmonic transform are also some other leading examples for discrete transforms, each of which has fast algorithms with reasonable computational complexities. For further details the reader is referred to Driscoll JR, Healy Jr, DM. & Rockmore DN (1997), Chen, W.H., Smith, C.H. and Fralick, S.C.(1977) and the references therein.

In this paper, we ourselves present a novel discrete transform. Let N be a given positive integer, and let $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ be a given sequence of N complex numbers.

For a given positive integer p and a complex parameter σ, we define our transform by the sequence

$$\alpha=\{\alpha_0(p),\alpha_1(p),\ldots,\alpha_{N-1}(p)\},$$

where

$$\alpha_k(p) = \sum_{j=0}^{N-1} \beta_j (\sigma + w^{jk})^p \qquad (1)$$

for all $k = 0,1,\ldots,N-1$.

It is not difficult to see that this transform with $\sigma = 0$ and $p = 1$ yields the DFT and that the naive computation of the transform requires $O(pN^2)$ arithmetic operations which is prohibitive for large values of N and p. We further develop a fast algorithm to compute the novel transform with computational complexity of $O(N \log N)$ for relatively small values of p, with respect to N. In this paper, the novel transform of the sequence $\{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$ will be denoted by $NDT(\beta)$ when needed. Also, the novel transform governs the properties of linearity and periodicity. To put it another way, let $\beta = \{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$, $\gamma = \{\gamma_0, \gamma_1, \ldots, \gamma_{N-1}\}$ be two sequences of complex numbers each containing N terms. Suppose further that a and b are two constants. Then,

1) **Linearity** :

$NDT(a\beta + b\gamma) = aNDT(\beta) + bNDT(\gamma)$

That is, the NDT is a linear operator.

2) **Periodicity**:

$\alpha_{k+N}(p) = \alpha_k(p)$

for all $k = 0,1,\ldots,N-1$.

The proof of the above two properties is obvious and thus we leave it.

The organization of this paper is set to have the following structure: In Section II, we present the development of a fast algorithm for the NDT. Section III describes the implementation of our algorithm. Section IV is devoted to present the numerical results. Section V makes comments on the numerical results. Finally, we give conclusions in Section VI.

II. Development of a Fast Algorithm for NDT

Since the great descriptions of the DFT and FFT are abundantly and frequently available in the literature, we ignore the description of those transforms, but we summarize both in the following theorem.

A. Theorem 1

Let $\{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$ be a sequence of N complex numbers. Then, the sequence of complex numbers $\{\alpha_0, \alpha_1, \ldots, \alpha_{N-1}\}$, defined by

$$\alpha_k = \sum_{j=0}^{N-1} \beta_j w^{jk}$$

for all $k = 0,1,\ldots,N-1$,

can be computed in $O(N \log N)$ operations.

The poof of Theorem 1 can be found in Cooley JW & Tukey JW (1965).

B. Proposition 1

Let $\{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$ be a sequence of N complex numbers, and let p be a positive integer such that N is divisible by p. Then the sequence of complex numbers $\{\alpha_0, \alpha_1, \ldots, \alpha_{N-1}\}$, defined by

$$\alpha_k = \sum_{j=0}^{N-1} \beta_j w^{jkp}$$

for all $k = 0,1,2,\ldots,N-1$,

can be computed in $O(N \log N)$ in operations.

Proof:

$$\alpha_k = \sum_{j=0}^{N-1} \beta_j w^{jkp} = \sum_{j=0}^{N-1} \beta_j e^{-i2\pi jkp/N} = \sum_{j=0}^{N-1} \beta_j e^{-i2\pi jk/N_p},$$

where $N_p = \frac{N}{p}$.

It can be easily seen that

$$e^{-i2\pi k(j+mN_p)/N_p} = e^{-i2\pi jk/N_p}$$

for all $j = 0,1,\ldots,N-1$,

where m is a non-negative integer.

Now,

$$\sum_{j=0}^{N-1} \beta_j e^{-2\pi jk/N_p} = \sum_{m=0}^{p-1} \sum_{j=0}^{N_p-1} \beta_{j+mN_p} e^{-i2\pi jk/N_p}$$

for all $k = 0,1,\ldots,N-1$.

We can further note that for each non-negative integer $q = 0,1,\ldots,p-1$,

$$\alpha_{k+qN_p} = \sum_{m=0}^{p-1} \sum_{j=0}^{N_p-1} \beta_{j+mN_p} e^{-i2\pi j(k+qN_p)/N_p} = \sum_{m=0}^{p-1} \sum_{j=0}^{N_p-1} \beta_{j+mN_p} e^{-i2\pi jk/N_p}$$

Hence, this implies that

$$\alpha_{k+qN_p} = \alpha_k$$

for all $k = 0,1,\ldots,N_p-1$, $m = 0,1,\ldots,p`-1$, and $q = 0,1,\ldots,p-1$.

The inner sum of the above double sum yields an N_p point DFT for each $m = 0,1,\ldots,p-1$, each of which can be computed in $3N_p/2\log N_p$ operations, using Theorem 1.

Thus, the total number of operations needed to compute all such N_p point DFT's amounts to

$$\left(\frac{3N_p}{2}\log N_p\right)p = \frac{3N}{2}\log\left(\frac{N}{p}\right).$$

To operate the outer sum of the double sum, it is required to have $N(p-1)/p$ additions and thus the total number of operations needed to operate the whole double sum equals

$$T_N = \frac{3N}{2}\log\left(\frac{N}{p}\right)+\frac{N}{p}(p-1).$$

This means that to compute α_k for all $k = 0,1,\ldots,N-1$, there must be at least T_N operations which is asymptotically equal to $O(N\log N)$ and thus we conclude the proof of Proposition 1.

C. Theorem 2

Let $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ be a sequence of N complex numbers. Suppose that N is divisible by 2.

Then the sequence of N complex numbers: $\{\alpha_0(2),\alpha_1(2),\ldots,\alpha_{N-1}(2)\}$, defined by the transform

$$\alpha_k(2) = \sum_{j=0}^{N-1}\beta_j(\sigma + w^{jk})^2$$

for all $k = 0,1,\ldots,N-1$, where σ is a complex parameter, can be computed in $O(N\log N)$ arithmetic operations.

Proof:

$$\alpha_k(2) = \sum_{j=0}^{N-1}\beta_j(\sigma + w^{jk})^2$$

$$= \sigma^2\sum_{j=0}^{N-1}\beta_j + 2\sigma\sum_{j=0}^{N-1}\beta_j w^{jk} + \sum_{j=0}^{N-1}\beta_j w^{2jk}$$

From Theorem 1 and Proposition 1, we can determine that the number of operations needed to compute the first, the second, and the third expressions are respectively:

$N+1$, $3N/2\log N + N + 1$, and $3N/2\log(N/2) + N/2$.

Thus the total number of operations taken to compute the entire transforms amounts to

$$3N\log N + 2N - \frac{3N}{2}\log 2 + 2,$$

which establishes that the asymptotic complexity of the new transform with $p = 2$ is $O(N\log N)$.

D. Theorem 3

Let N be a positive integer such that N is divisible by 2 and 3. Let $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ be a sequence of N complex numbers.

Then the sequence of N complex numbers: $\{\alpha_0(3),\alpha_1(3),\ldots,\alpha_{N-1}(3)\}$, defined by the transform

$$\alpha_k(3) = \sum_{j=0}^{N-1}\beta_j(\sigma + w^{jk})^3$$

for all $k = 0,1,\ldots,N-1$, where σ is a complex parameter, can be computed in $O(N\log N)$ arithmetic operations.

Proof:

$$\alpha_k(3) = \sum_{j=0}^{N-1}\beta_j(\sigma + w^{jk})^3$$

$$= \sigma^3\sum_{j=0}^{N-1}\beta_j + 3\sigma^2\sum_{j=0}^{N-1}\beta_j w^{jk} + 3\sigma\sum_{j=0}^{N-1}\beta_j w^{2jk} + \sum_{j=0}^{N-1}\beta_j w^{3jk}$$

Proposition 2 can be used to compute the number of operations needed for each of the last three finite sums in the above expression. Then It can be shown that the total number of operations that we want to accomplish the $\alpha_k(3)$ for all $k = 0,\ldots,N$ amounts to

$$\frac{9N}{2}\log N - \frac{3N}{2}\log 6 + \frac{43N}{6} + 3.$$

Now,

$$\left|\frac{9N}{2}\log N - \frac{3N}{2}\log 6 + \frac{43N}{6} + 3\right|$$

$$< \frac{9N}{2}\log N + \frac{3N}{2}\log N + \frac{61N}{6}\log N$$

$$= \frac{85}{6}N\log N \text{ whenever } N > 6.$$

Thus this concludes the proof of Theorem 3.

E. Theorem 4

Let N be a given positive integer, and let $p(>1)$ be a positive integer such that N is divisible by each of $2,3,\ldots,p-1$.

Let $\{\beta_0,\beta_1,\ldots,\beta_{N-1}\}$ be a sequence of N complex numbers.

Then the sequence of N complex numbers: $\{\alpha_0(p),\alpha_1(p),\ldots,\alpha_{N-1}(p)\}$, defined by the transform

$$\alpha_k(p) = \sum_{j=0}^{N-1}\beta_j(\sigma + w^{jk})^p$$

for all $k = 0,1,\ldots,N-1$, where σ is a complex parameter, can be computed in $O(N\log N)$ arithmetic operations.

Proof:

Now let us consider the case $p > 2$.

$$\alpha_k(p) = \sum_{j=0}^{N-1} \beta_j (\sigma + w^{jk})^p$$
$$= \sum_{j=0}^{N-1} \beta_j \sum_{r=0}^{p} \binom{p}{r} \sigma^{p-r} w^{jkr}$$
$$= \sum_{r=0}^{p} \binom{p}{r} \sigma^{p-r} \sum_{j=0}^{N-1} \beta_j w^{jkr}$$
$$= \sigma^p \sum_{j=0}^{N-1} \beta_j + \sum_{r=1}^{p} \binom{p}{r} \sigma^{p-r} \sum_{j=0}^{N-1} \beta_j w^{jkr},$$

where $\binom{p}{r} = \frac{p!}{(p-r)!r!}$.

To accomplish the first sum together with term σ^p, we want $N + p - 1$ operations.

Then, from Proposition 2, for a fixed $r(>0)$, the number of operations taken to compute $\sum_{j=0}^{N-1} \beta_j w^{jkr}$ is

$$\frac{3N}{2}\log\left(\frac{N}{r}\right) + \frac{N}{r}(r-1)$$

while the number of operations taken to compute the term $\binom{p}{r}\sigma^{p-r}$ is $2p - r + 1$ for $r > 0$. Further, we want additional Np operations to add all the individual sums together for all $k = 0,1,\ldots,N-1$.

Thus the total number of operations needed to compute the new transform is

$$\sum_{r=1}^{p}\left(\frac{3N}{2}\log\left(\frac{N}{r}\right) + \frac{N}{r}(r-1) + 2p - r + 1\right) + N + p - 1 + Np$$
$$= \frac{3N}{2}\left(p\log N - \log p!\right) - N\sum_{r=1}^{p}\left(\frac{1}{r}\right) + \left(\frac{3}{2}p^2 + \frac{3}{2}p + 2Np + N - 1\right)$$

Hence,

$$\left|\sum_{r=1}^{p}\left(\frac{3N}{2}\log\left(\frac{N}{r}\right) + \frac{N}{r}(r-1) + 2p - r + 2\right) + N + p - 1\right|$$
$$< \frac{3Np}{2}\log N + \frac{3Np}{2}\log N + pN\log N + \frac{7Np}{2}\log N$$
$$= \frac{15p}{2}N\log N$$

whenever $N > \max(p+1, p!)$.

For relatively small values of p with respect to N, the computational complexity of the transform is $O(N\log N)$.

III. IMPLEMENTATION

This section concentrates on the implementation of Theorem 2, Theorem 3 and Theorem 4 described in Section III.

Algorithm 1: Theorem 2

INPUT: $\sigma, w, \beta = \{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$

OUPUT: $NDT(\beta): \{\alpha_0(2), \alpha_1(2), \ldots, \alpha_{N-1}(2)\}$

STAGES:

1. Compute $c = \sigma^2(\beta_0 + \beta_1 + \ldots + \beta_{N-1})$
   Set $C = [c_0, c_1, \ldots, c_{N-1}]^{\mathrm{T}}$,
   where $c_l = c$ for all $l = 0,1,\ldots,N-1$.
2. Let :
   $\pi_{11} = [\beta_0, \beta_1, \ldots, \beta_{N-1}]^{\mathrm{T}}$
   $\pi_{21} = [\beta_0, \beta_1, \ldots, \beta_{N/2-1}]^{\mathrm{T}}$
   $\pi_{22} = [\beta_{N/2}, \beta_{N/2+1}, \ldots, \beta_{N-1}]^{\mathrm{T}}$
   Compute:
   a) $D_{11} = 2\sigma FFT(\pi_{11})$
   b) $D_{21} = FFT(\pi_{21})$
   c) $D_{22} = FFT(\pi_{22})$
3. Compute:
   $$NDT(\beta) = C + D_{11} + \begin{bmatrix} D_{11} \\ D_{11} \end{bmatrix} + \begin{bmatrix} D_{22} \\ D_{22} \end{bmatrix}$$

Algorithm 2: Theorem 3

INPUT: $\sigma, w, \beta = \{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$

OUPUT: $NDT(\beta): \{\alpha_0(3), \alpha_1(3), \ldots, \alpha_{N-1}(3)\}$

STAGES:

1. Compute $c = \sigma^3(\beta_0 + \beta_1 + \ldots + \beta_{N-1})$
   Set $C = [c_0, c_1, \ldots, c_{N-1}]^{\mathrm{T}}$,
   where $c_l = c$ for all $l = 0,1,\ldots,N-1$.
2. Let:
   $\pi_{11} = [\beta_0, \beta_1, \ldots, \beta_{N-1}]^{\mathrm{T}}$
   $\pi_{21} = [\beta_0, \beta_1, \ldots, \beta_{N/2-1}]^{\mathrm{T}}$
   $\pi_{22} = [\beta_{N/2}, \beta_{N/2+1}, \ldots, \beta_{N-1}]^{\mathrm{T}}$
   $\pi_{31} = [\beta_0, \beta_1, \ldots, \beta_{N/3-1}]^{\mathrm{T}}$
   $\pi_{32} = [\beta_{N/3}, \beta_{N/3+1}, \ldots, \beta_{2N/3-1}]^{\mathrm{T}}$
   $\pi_{33} = [\beta_{2N/3}, \beta_{2N/3+1}, \ldots, \beta_{N-1}]^{\mathrm{T}}$
   Compute:
   a) $D_{11} = 3\sigma^2 FFT(\pi_{11})$
   b) $D_{21} = 3\sigma FFT(\pi_{21})$
   c) $D_{22} = 3\sigma FFT(\pi_{22})$
   d) $D_{31} = FFT(\pi_{31})$

e) $D_{32,} = FFT(\pi_{32})$

f) $D_{33} = FFT(\pi_{33})$

3. Compute:

$$NDT(\beta) = C + D_{11} + \begin{bmatrix} D_{21} \\ D_{21} \end{bmatrix} + \begin{bmatrix} D_{22} \\ D_{22} \end{bmatrix} + \begin{bmatrix} D_{31} \\ D_{31} \\ D_{31} \end{bmatrix} + \begin{bmatrix} D_{32} \\ D_{32} \\ D_{32} \end{bmatrix} + \begin{bmatrix} D_{33} \\ D_{33} \\ D_{33} \end{bmatrix}$$

Algorithm 3: Theorem 4

INPUT: $\sigma, w, p, \beta = \{\beta_0, \beta_1, \ldots, \beta_{N-1}\}$

OUPUT: $NDT(\beta) : \{\alpha_0(p), \alpha_1(p), ..., \alpha_{N-1}(p)\}$

STAGES:

1. Compute $c = \sigma^p(\beta_0 + \beta_1 + \ldots + \beta_{N-1})$

   Set $C = [c_0, c_1, \ldots, c_{N-1}]^{\mathrm{T}}$,

   where $c_l = c$ for all $l = 0,1,\ldots,N-1$.

2. Let:

   $\pi_{11} = [\beta_0, \beta_1, \ldots, \beta_{N-1}]^{\mathrm{T}}$

   $\pi_{mn} = [\beta_{N(m-1)/p}, \beta_{N(m-1)/p+1}, \ldots, \beta_{mN/p-1}]^{\mathrm{T}}$

   for all $m = 1,2,\ldots,p$ and for $n = 1$ to m

   Compute:

   a) $D_{11} = \binom{p}{1}\sigma^{p-1} FFT(\pi_{11})$

   b) for $m = 2$ to do p

   for n =1 to do m

   $$D_{mn} = \binom{p}{m}\sigma^{p-m} FFT(\pi_{mn})$$

   $$D_{mn}^m = [D_{mn}, D_{mn}, \ldots, D_{mn}]^{\mathrm{T}}$$

   (This includes m components)

   end

   end

3. Compute:

   for $m = 2$ to do p

   $$NDT(\beta) = C + D_{11} + \sum_{n=1}^{m} D_{mn}^m$$

   end

IV. NUMERICAL RESULTS

This section shows numerical results of numerical experiments, carried out to illustrate the performance of the novel algorithm. We present numerical tests for $p = 2$ and $p = 3$. In each case, the sequence β was randomly chosen from the interval (0, 1) by operating MATLAB rand() function for various values of N, and further the *fft*() function in MATLAB was used to efficiently compute all the DFT's involved in the fast algorithm. All the computations were performed on a personal computer with Intel(R) Pentium 2.1 GHz processor, with 2.00 GB RAM, 64 bit windows 7 operating system using MATLAB version 12 codes. The efficiency of the fast transform is tested by means of CPU times in seconds (sec.). The accuracy of the algorithm is tested with respect to relative errors (RE). The relative error of the output sequence α is computed with respect to the maximum relative error (MRE) and the infinity norm defined by (2) and (3) , respectively. In the error formulae, v and v^* stand for the result computed by the new fast transform and the corresponding result computed by the naive algorithm, respectively.

$$\mathrm{MRE} = \max_{i=0,1,\ldots,n-1} \frac{\left|v_i - v_i^*\right|}{\left|v_i^*\right|} \tag{2}$$

$$RE_\infty = \frac{\max_{0\le i\le n-1}\left|v_i - v_i^*\right|}{\max_{0\le i\le n-1}\left|v_i^*\right|} \tag{3}$$

Table 1: Relative error when $p = 2$ and $\sigma = 10$

N	MRE	RE_∞
128	8.323e-15	7.547e-15
256	1.463e-14	1.326e-14
512	2.136e-13	1.949e-13
1024	4.140e-13	3.756e-13
2048	2.899e-12	2.629e-12
4096	9.465e-12	8.577e-12
8192	1.910e-11	1.734e-11

Table 2: Relative error when $p = 3$ and $\sigma = 10$

N	*MRE*	RE_∞
1566	4.998e-12	3.748e-12
1686	6.041e-12	4.569e-12
1806	8.446e-12	6.373e-12
2106	1.363e-11	1.024e-11
2706	2.790e-11	2..096e-11
3006	1.888e-11	1.423e-11
3606	3.070e-13	2.301e-13
4206	6.053e-11	4.562e-11
4806	3.306e-11	2.484e-11
5406	1.884e-11	1.420e-11
6006	5.377e-11	4.035e-11
6606	1.072e-10	4.035e-11
7206	7.514e-12	5.648e-12
7806	1.402e-10	1.054e-10
8406	1.996e-10	1.497e-10
9006	8.242e-12	6.182e-12
10206	7.063e-11	5.306e-11
10806	2.963e-11	2.229e-11

Table 3: CPU times elapsed for the fast algorithm when $p = 3$ $\sigma = 10$, and $N \geq$ 12006

N	12006	15006	18006	21006	24006	30006	36006	48006
CPU times	0.312	0.359	0.515	0.702	0.889	1.357	1.966	3.494
N	48006	60006	72006	90006	120006	150006	180006	240006
CPU times	3.494	5.351	7.691	12.870	22.511	35.319	52.058	98.031
N	300006	600006	1200006					
CPU times	158.653	654.019	2564.000					

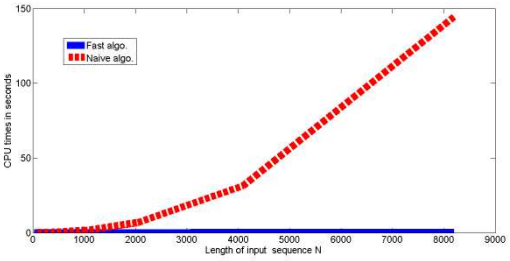

Figure 1: Comparison of CPU times between the NAIVE AND FAST ALGORITHMS WHEN $p = 2,\ \sigma = 10$

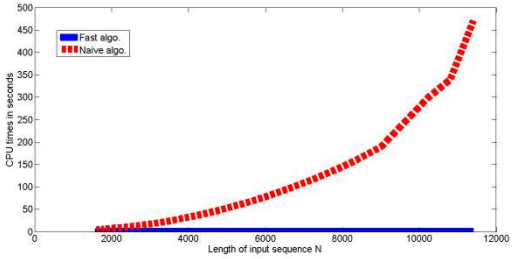

Figure 2: Comparison of CPU times between the naive and fast algorithms when $p = 3, \sigma = 10$

V. DISCUSSION

Figure 1 demonstrates the comparison of CPU times for the naive and fast algorithms for different values of N when $p = 2$ and $\sigma = 10$. In this case $p = 2$, the

values being chosen for N must be divisible by 2(see Theorem 2). In this example, we have chosen the values 2, 4, 8, ..., 4096, 8192 for N. It can be seen from

Figure 1 that CPU times elapsed for the fast algorithm is comparatively very smaller than those for the naive algorithm. At $N = 128$, CPU times for both algorithms are almost the same (0.000 seconds), whereas at $N =$ 8192, the CPU time for the naive algorithm is nearly 150 seconds and so is for the fast algorithm is 0.000 seconds. Figure 2 illustrates the comparison of CPU times for both algorithms in the case where $p = 3$ and $\sigma = 10$. As this illustration includes p =3, the values being chosen for N must be divisible by both 2 and 3(see Theorem 3). In this case, the values chosen for N are 1566, 1686, 1806, 2106, 2706, 3006, 3606, 4206, 4806, 5406, 6006, 7206, 7806, 8406, 9006, 10206, 10806, 11406(the values of N in which $N < 1566$ are ignored since the CPU times for the fast algorithm are 0.000). Figure 2 also exhibits a relationship between CPU times elapsed for the fast algorithm and the CPU times elapsed for the naive algorithm which is similar to that in Figure 1(At N =11406, the CPU time for the naive algorithm is nearly 450 seconds whereas that for the fast algorithm is 0,000). On the other hand, the computer memory capacity is inadequate for the naive algorithm to be implemented when N >12006. The CPU times elapsed for the fast algorithm when $N \geq 12006$ are included in Table 3. It displays that when N =12006, the CPU time is 0.312 seconds, while when N =1200006, the CPU time is 2564 seconds. Table 1 and Table 2 shows the

relative errors, namely *MRE* and RE_∞ computed at various values of N. These errors confirm that the fast algorithm permits high accuracy to be attained. In Table 1 MRE=8.323e-15 and RE_∞ =7.547e-15 when $N = 128$, while MRE=1.910e-11 and RE_∞ =1.734e-11 when $N = 8192$. Besides that in Table 1, *MRE*=4.998e-12 and RE_∞ =3.748e-12 when N =1566 and, MRE = 2.383e-10 and RE_∞ =1.791e-10 when N =11406.

VI. CONCLUSION

In this paper, we defined a novel discrete transform and developed a fast algorithm for it. For relatively small values of p, the computational cost of the fast algorithm is $O(N \log N)$. The theoretical results for $p = 2,3$ were also numerically confirmed. It is also possible to verify the theoretical results for $p > 3$. All the numerical computations were performed in MATLAB. We have further shown that the novel transform holds the properties of linearity and periodicity. However, the existence of the inversion of the transform is yet to be determined. The inversion, some useful active applications of the transform and the fractional form of the transform would be the subject of the further work.

REFERENCES

Cooley JW. & Tukey JW (1965). "An algorithm for the machine calculation of complex Fourier series," *Mathematics of computation*, vol. 19, 297- 301pp.

Driscoll JR , Healy Jr, DM. & Rockmore DN (1997). "Fast discrete polynomial transform with applications to data analysis for distance transitive graphs," *SIAM Journal on Computing*, vol.26, 1066-1099 pp.

Chen, W.H., Smith, C.H. and Fralick, S.C., 1977. A fast computational algorithm for the discrete cosine transform. *IEEE Transactions on communications*, *25*(9), pp.1004-1009.

BIOGRAPHY OF AUTHORS

WA Gunarathna is a mathematics lecturer at the department of Mathematics of Faculty of Engineering, General Sir John Kotelawala Defence University, Ratmalana, Sri Lanka. His current research interests include design of super fast algorithms for discrete transforms, numerical solutions of fractional order differential equations, developing fast and robust algorithms for the Quadratic eigenvalue problem(QEP), Computational Number theory and Computational Algebra.

Nasir HM is a Visiting Consultant at the department of Mathematics and Statistics of College of Science of Sultan Qaboos University, Oman . He received Master of Engineering (M.Eng) and Ph.D in computational Mathematics, both from the University of Electro Communications, Tokyo, JAPAN in 1999 and 2003, respectively. His research interests include Numerical solutions of partial differential equations, and multi complex analysis