Reducing the multiplication count in *m*-dimensional FFT by a factor *m* with classical methods

Þorgeir Sigurðsson, Stefán I. Valdimarsson, Sven Þ. Sigurðsson

***Abstract*— We propose a straightforward extension of the Cooley-Tukey Fast Fourier Transform algorithm to handle *m*-dimensional data. Our algorithm reduces the number of multiplications needed when compared with the row-column method and the vector transform method without increasing the number of additions.**

The decrease in the number of complex multiplications when compared with the row-column method is asymptotically by a factor *m*, as if multiplications were only used when transforming one of the *m* dimensions.

The recursion relations which arise when analyzing the number of operations for the new algorithm have a number of interesting properties not seen before in the study of Fast Fourier Transforms.

***Index Terms*—Fast Fourier Transforms, Signal Processing Algorithms, Computational Efficiency, Multivariate Recurrences**

I. Introduction

When comparing different approaches to transforming m-dimensional data the situation appears still to be as described more than 20 years ago in a 1990 tutorial by Duhamel & Vetterli [1] which may be paraphrased as follows: The row-column algorithm is the one allowing the easiest implementation and it can easily be parallelized. Vector transforms have a lower number of arithmetic operations but they are more complex. Polynomial transforms have a still lower number of arithmetic operations but little work has been done on finding the best way of implementing them.

Here we present a new algorithm for multi-dimensional Discrete Fourier Transforms (DFT). It is an extension of the one dimensional Cooley-Tukey Fast Fourier Transform (FFT) and may both be seen as a variation and an improvement of the row-column and the Vector transform algorithms. Its performance in terms of arithmetic complexity is comparable to that of the structurally more complex Polynomial transform of Nussbaumer and Quandalle [2] and the method of Bernardini [3] which is based on periodicity lattices.

In the new algorithm we work with all dimensions simultaneously and try as much as possible to collect multiplications across the dimensions. This is similar to the idea behind the vector transform but we do the collection more efficiently. We call the new algorithm the *Diagonal FFT*.

The algorithm is of a general nature and can be adapted for all radices and also the split radix method. To simplify the exposition we focus on the radix-2, radix-4 and the split radix cases in the present paper.

Table 1 compares the number of complex multiplications in the diagonal method with those in the row-column method and the vector transform methods. With m dimensions and $N = (2^k)^m$ approaching infinity, the dominant term in the number of multiplications is $aN\lg N$ with the constant a listed in the table.

TABLE 1
Value of the constant a in the dominant term $aN\lg N$ of complex multiplications for very large FFTs.

Algorithm	2D	3D	*m-dimensions*
Row-Column Radix 2	0.500	0.500	1/2
Row-Column Radix 4	0.375	0.375	3/8
Row-Column Split Radix	0.333	0.333	1/3
Vector Radix 2	0.375	0.292	$(1/m)(1-2^{-m})$
Vector Radix 4	0.234	0.164	$1/(2m)(1-4^{-m})$
Vector Split Radix	0.214	0.156	$\frac{1}{2m}\cdot\frac{2^{m+1}-2}{2^{m+1}-1}$
Diagonal Radix 2	0.250	0.167	$1/(2m)$[a]
Diagonal Radix 4	0.188	0.125	$3/(8m)$[a]
Diagonal Split Radix	0.167	0.111	$1/(3m)$[a]

The formulae for the Vector radix 2 and Vector radix 4 algorithms can be found in [4] and [5]. The Vector Split Radix algorithm is the one presented in [1] while it has also other varieties with different performances [5].

The number of complex additions, not associated with multiplications, is $N\lg N$, the same for all the algorithms in *Table 1*.

Transforms that reduce the number of multiplications by a factor $1/m$ over a row-column approach have been proposed, in

Manuscript received October 15, 2013.

Þorgeir Sigurðsson is with the Icelandic Radiation Safety Authority, 150 Reykjavik, Iceland (e-mail: ts@gr.is).

Stefán I. Valdimarsson is with the Division of Mathematics, Science Institute, University of Iceland, Dunhaga 5, 107 Reykjavik, Iceland (e-mail: siv@hi.is).

Sven Þ. Sigurðsson is with the Faculty of Industrial Engineering, Mechanical Engineering and Computer Science, University of Iceland, Dunhaga 5, 107 Reykjavik, Iceland (e-mail: sven@hi.is).

[a] For $m > 2$ this is a conjecture, see section III.D for supporting evidence.

particular the polynomial transform of Nussbaumer and Quandalle [2] when the sizes of all dimensions are equal, see also [4] where it is stated that this is the best known algorithm for this case, and the more general periodicity lattice approach of Bernardini [3]. These thus have the same asymptotic multiplication count as the Diagonal transform. For finite values of *m* the multiplication count for these transforms can be lower but they have drawbacks in other respects. The method of Nussbaumer and Quandalle uses three permutation steps (bit-reversals) compared with one permutation step for the Diagonal FFT and other algorithms of Cooley-Tukey type. The method of Bernardini also uses extra permutations. The result is that the total operation count, including multiplications, additions and assignments, is smallest for the diagonal method. This is discussed further in Section III.C.

The Diagonal transform can easily be parallelized. As in the one dimensional Cooley-Tukey FFT, the data is recursively split into two parts which are operated on independently. Thus the potential for parallel operations and for operations that only need to access data points not far apart is similar to the one dimensional Cooley-Tukey situation.

In section II we explain the diagonal method by applying it to the classic radix-2 FFT. In section III we give some results for the number of multiplications the method uses in two dimensions, both theoretical and empirical, in the case of radix-4 and split-radix, as well as radix-2. Especially important is our observation that in two dimensions the next term following the dominant term is $O(N\sqrt{\lg N})$ while it is only $O(N)$ for the row-column and vector transforms. Practical and theoretical consequences of this are discussed.

II. Description of the Algorithm

A. Radix 2

The Diagonal FFT is based on the Cooley-Tukey method. For one dimensional data of even length $T_N = (x_i)_{i=0}^{N}$ we write the discrete Fourier transform (DFT) $(X_k)_{k=0}^{N}$ of the data as

$$X_k = \sum_{i=0}^{N/2-1} x_{2i}\omega_{N/2}^{ik} + \omega_N^k \sum_{i=0}^{N/2-1} x_{2i+1}\omega_{N/2}^{ik} \tag{1}$$

where $\omega_N = e^{-j2\pi N}$ is a primitive *N*-th root of unity.

The first summation expresses the *DFT* of the even-indexed elements of the input vector x_i. The second summation expresses the *DFT* of the odd-indexed elements. One additional complex multiplication is needed for each of the *N*/2 elements of the latter *DFT* output vector, compared with the former vector. Finally we note that $X_{N/2+k}$ has exactly the same expression as X_k except there is now a minus sign between the two terms.

We write (1) as:

$$\begin{aligned} DFT[T_N](k) &= DFT[T_{N/2}](k) + DFTo[To_{N/2}](k), \\ DFT[T_N](N/2+k) & \\ &= DFT[T_{N/2}](k) \\ &\quad - DFTo[To_{N/2}](k). \end{aligned} \tag{2}$$

Here $T_{N/2}$ and $To_{N/2}$ denote the even-indexed and odd-indexed elements of T_N respectively and *DFTo* denotes the transform in the second term of (1).

The *DFTo* is related to the ordinary *DFT* through (3) which expresses the well known Shift theorem.

$$DFTo[To_{N/2}](k) = \omega_N^k DFT[To_{N/2}](k) \tag{3}$$

In the row-column method, equations (2) and (3) are applied recursively in the first dimension and then their equivalents are applied successively for each of the other remaining dimensions. In our method we postpone the multiplications in (3) (one complex multiplication for each $k = 0, \dots, N/2 - 1$) and work instead immediately with a new dimension.

We start by rewriting (2) for a T that is a *m*-dimensional matrix of size $N_1 \times N_2 \ldots \times N_m$.

$$\begin{aligned} &DFT[T_{N_1,N_2,\dots,N_m}](k_1,k_2,\dots,k_m) \\ &\qquad = DFT[T_{N_1/2,N_2,\dots,N_m}] \\ &\qquad + DFTo_1\left[To_{1_{N_1/2,N_2,\dots,N_m}}\right], \\ &DFT[T_{N_1,N_2,\dots,N_m}](N_1/2+k_1,k_2,\dots,k_m) \\ &\qquad = DFT[T_{N_1/2,N_2,\dots,N_m}] \\ &\qquad - DFTo_1\left[To_{1_{N_1/2,N_2,\dots,N_m}}\right]. \end{aligned} \tag{4}$$

Here, and in what follows, we omit (k_1, k_2, … k_m) from the right hand sides of our equations.

We decompose $DFT[T_{N_1/2,N_2,\dots,N_m}]$ further using (4) again but we decompose *DFTo*₁ into even and odd sub-transforms along the second dimension as follows.

$$\begin{aligned} &DFTo_1\left[To_{1_{N_1/2,N_2,\dots,N_m}}\right](k_1,k_2,\dots,k_m) \\ &= DFTo_1\left[To_{1_{N_1/2,N_2/2,N_3,\dots,N_m}}\right] \\ &+ DFTo_1o_2\left[To_1o_{2_{N_1/2,N_2/2,N_3,\dots,N_m}}\right], \\ &DFTo_1\left[To_{1_{N_1/2,N_2,\dots,N_m}}\right](k_1,N_2/2+k_2,\dots,k_m) \\ &= DFTo_1\left[To_{1_{N_1/2,N_2/2,N_3,\dots,N_m}}\right] \\ &- DFTo_1o_2\left[To_1o_{2_{N_1/2,N_2/2,N_3,\dots,N_m}}\right]. \end{aligned} \tag{5}$$

Here we have introduced the notation $DFTo_1o_2$ for the transform of To_1o_2, the part of To_1 with odd indices in the second dimension.

If there are more than two dimensions we keep postponing any multiplications and continue dividing the matrix into even and odd transforms along the third dimension and then the fourth, etc, until we reach the final dimension *m*.

For $DFTo_1o_2 \dots o_m$ we have:

$$\begin{aligned} &DFTo_1o_2 \dots o_m[To_1o_2 \dots o_m] \\ &\qquad = \omega_{N_1}^{k_1}\omega_{N_2}^{k_2} \dots \omega_{N_m}^{k_m} DFT[To_1o_2 \dots o_m] \end{aligned} \tag{6}$$

The transforms $DFTo_1o_2 \dots o_m$ and DFT are related through a shift along a multidimensional diagonal, expressed by (6).

The boundary cases for this decomposition are when $N_i = 1$ for some *i*. When that happens we eliminate the *i*-th dimension from the indexing, For example a three dimensional array with dimensions of sizes $(N_1, 1, N_3)$ is equivalent to a two dimensional array with dimensions of sizes (N_1, N_3).

We note that the recursive structure of the Diagonal FFT is somewhat different from the structure for the row-column and the vector transform FFTs. This is because we use different paths to handle the even and odd cases. For example in three dimensions, To_1o_2 is handled by $DFTo_1o_2$ which breaks it into the even-indexed and the odd-indexed part in the third dimension. The even-indexed part is again handled by $DFTo_1o_2$ whereas the odd-indexed part is handled by $DFTo_1o_2o_3$. In particular this means that the Diagonal FFT cannot be described in terms of *I/O tensors* which are the building blocks of the algorithms implemented in the popular FFT system FFTW designed by Frigo and Johnson [6].

B. Split Radix and other radices

In the split radix algorithm, more precisely the conjugate pair FFT [7] and [8], we split the second term in (1) into two pieces

$$X_k = \sum_{i=0}^{N/2-1} x_{2i}\omega_{N/2}^{ik} + \omega_N^k \sum_{i=0}^{N/4-1} x_{4i+1}\omega_{N/4}^{ik} + \omega_N^{-k} \sum_{i=0}^{N/4-1} x_{4i-1}\omega_{N/4}^{ik} \tag{7}$$

The first summation expresses the *DFT* of the even-indexed elements ($T_{N/2}$) of the input vector T_N. The second summation expresses the *DFT* of half of the remaining elements with indices $4i+1$ and beginning with x_1, the third is the *DFT* of the remaining quarter with indices $4i$-1 beginning with x_{-1} (equal to x_{N-1}). One complex multiplication is needed for each of the N/4 elements of the two latter *DFT*s. This formula is used directly to calculate X_k for k less than N/4 and using that $\omega_N^{N/4+k} = j\omega_N^k$ we can see that the rest of X_k can be calculated without further multiplications.

We write equation (7) as

$$\begin{aligned} DFT\left[T_{N_1,N_2,\dots,N_m}\right] &= DFT\left[T_{N_1/2,N_2,\dots,N_m}\right] \\ &+ DFToe_1\left[Toe_{1_{N_1/4,N_2,\dots,N_m}}\right] \\ &+ DFToo_1\left[Too_{1_{N_1/4,N_2,\dots,N_m}}\right] \end{aligned} \tag{8}$$

where we have introduced the notation Toe_1 and Too_1 for the $4j+1$ and the $4j$-1 indexed elements respectively.

Then we proceed as before, $DFT\left[T_{N_1/2,N_2,\dots,N_m}\right]$ we decompose further using (8) again but we decompose $DFToe_1\left[Toe_{1_{N_1/4,N_2,\dots,N_m}}\right]$ along the second dimension as follows

$$\begin{aligned} &DFToe_1\left[Toe_{1_{N_1/4,N_2,\dots,N_m}}\right] \\ &= DFToe_1\left[Toe_{1_{N_1/4,N_2/2,\dots,N_m}}\right] \\ &+ DFToe_1oe_2\left[Toe_1oe_{2_{N_1/4,N_2/4,\dots,N_m}}\right] \\ &+ DFToe_1oo_2\left[Toe_1oo_{2_{N_1/4,N_2/4,\dots,N_m}}\right] \end{aligned} \tag{9}$$

and similarly for $DFToo_1\left[Too_{N_1/4,N_2,\dots,N_m}\right]$. We use decompositions like these in each dimension and in the final, m-th, dimension we do the multiplications, for example

$$\begin{aligned} &DFToe_1oe_2 \dots oe_m[Toe_1oe_2 \dots oe_m] \\ &= \omega_{N_1}^{k_1}\omega_{N_2}^{k_2} \dots \omega_{N_m}^{k_m} DFT[Toe_1oe_2 \dots oe_m]. \end{aligned} \tag{6}$$

The boundary cases are when $N_i = 1$ or $N_i = 2$ for some i. The case $N_i = 1$ we handle as before and the $N_i = 2$ case is similar since two point DFTs do not require any multiplications.

Adapting the Diagonal method to other one dimensional algorithms of Cooley-Tukey type, e.g. using other radices, is similarly straightforward.

C. An implementation in two dimensions

A recursive implementation of the 2D Diagonal Radix 2 FFT is outlined below. We assume N is equal to $N_1 \times N_2$ where both N_1 and N_2 are integer powers of 2 and we construct the subroutines DFT, $DFTo_1$ and $DFTo_1o_2$ that perform the operations described by (2), (5) and (6).

```
function $DFT(x_{i,j}, N_1, N_2)$
  if $N_1 > 1$ then
                              → Use DFT to transform the even-numbered rows
    $u_{i=0,\dots,N_1/2-1,j=0,\dots,N_2-1} \leftarrow DFT(x_{i=0,2,\dots,N_1-2,j=0,\dots,N_2-1}, N_1/2, N_2)$
                              → Use DFTo to transform the odd-numbered rows
    $v_{i=0,\dots,N_1/2-1,j=0,\dots,N_2-1} \leftarrow DFTo_1(x_{i=1,3,\dots,N_1-1,j=0,\dots,N_2-1}, N_1/2, N_2)$
    for all $i = 0,\dots,N_1/2-1$, $j = 0,\dots,N_2-1$ do
                                                   → Combine with a butterfly
      $y_{i,j} \leftarrow u_{i,j} + v_{i,j}$
      $y_{i+N_1/2,j} \leftarrow u_{i,j} - v_{i,j}$
    end for
  else
                                     → Use 1D FFT to transform the single row
    $y_{i=0,j=0,\dots,N_2-1} \leftarrow DFT1d(x_{i=0,j=0,\dots,N_2-1}, N_2)$
  end if
  return $y$
end function

function $DFTo_1(x_{i,j}, N_1, N_2)$
  if $N_2 > 1$ then
                      → Use $DFTo_1$ to transform the even-numbered columns
    $u_{i=0,\dots,N_1-1,j=0,\dots,N_2/2-1} \leftarrow DFTo_1(x_{i=0,\dots,N_1-1,j=0,2,\dots,N_2-2}, N_1, N_2/2)$
                      → Use $DFTo_1o_2$ to transform the odd-numbered columns
    $v_{i=0,\dots,N_1-1,j=0,\dots,N_2/2-1} \leftarrow DFTo_1o_2(x_{i=0,\dots,N_1-1,j=1,3,\dots,N_2-1}, N_1, N_2/2)$
    for all $i = 0,\dots,N_1-1$, $j = 0,\dots,N_2/2-1$ do
                                                   → Combine with a butterfly
      $y_{i,j} \leftarrow u_{i,j} + v_{i,j}$
      $y_{i,j+N_2/2} \leftarrow u_{i,j} - v_{i,j}$
    end for
  else
                                  → Use 1D FFT to transform the single column
    $y_{i=0,\dots,N_1-1,j=0} \leftarrow DFT1d(x_{i=0,\dots,N_1-1,j=0}, N_1)$
    for all $i = 0,\dots,N_1-1$ do
                                               → Multiply by twiddle factors
      $y_{i,0} \leftarrow \omega_{2N_1}^{i} y_{i,0}$
    end for
  end if
  return $y$
end function

function $DFTo_1o_2(x_{i,j}, N_1, N_2)$
  $y \leftarrow DFT(x, N_1, N_2)$          → Use DFT to transform the matrix
  for all $i = 0,\dots,N_1-1$, $j = 0,\dots,N_2-1$ do
                                               → Multiply by twiddle factors
    $y_{i,j} \leftarrow \omega_{2N_1}^{i}\omega_{2N_2}^{j} y_{i,j}$      → This is done with one multiplication
  end for
  return $y$
end function
```

III. Complexity Analysis

A. Complex Multiplications for Radix 2

We seek an expression for the number of complex multiplications for the Radix 2 Diagonal FFT transform in two dimensions. This is a somewhat arbitrary measure since multiplications by ω_8 and powers of it use fewer real multiplications than multiplications by general complex numbers. However, by studying the complex multiplication count we get a good indicator of the properties of the real multiplication count without too many technical difficulties.

We assume $N_1 = 2^{k_1}$ and $N_2 = 2^{k_2}$ and let $M(k_1, k_2)$, $Mo(k_1, k_2)$ and $Moo(k_1, k_2)$ denote the number of complex multiplications needed for calculating DFT, $DFTo_1$, and $DFTo_1o_2$, respectively, of a matrix of size $N_1 \times N_2$. By examining equations (2), (5) and (6) we note the following relationships

(7) $$M(k_1,k_2) = M(k_1-1,k_2) + Mo(k_1-1,k_2),$$

(8) $$Mo(k_1-1,k_2) = Mo(k_1-1,k_2-1) + Moo(k_1-1,k_2-1),$$

and

(9) $$Moo(k_1-1,k_2-1) = M(k_1-1,k_2-1) + 2^{k_1-1}2^{k_2-1}.$$

By inserting Moo from (9) into (8) and the resulting right hand side into (7) we get

(10) $$M(k_1,k_2) = M(k_1-1,k_2) + Mo(k_1-1,k_2-1) + M(k_1-1,k_2-1) + 2^{k_1-1}2^{k_2-1}.$$

Finally by replacing k_2 with $k_2 - 1$ in (7) and inserting into (10) we get

(11) $$M(k_1,k_2) = M(k_1-1,k_2) + M(k_1,k_2-1) + 2^{k_1-1}2^{k_2-1}.$$

In the boundary cases, when either N_1 or N_2 is equal to 1, we use the one dimensional radix 2 FFT so that

(12) $$M(k,0) = M(0,k) = k2^{k-1} = N\lg(N)/2.$$

The equation system (11) and (12) is a recurrence relation of a kind that has not been seen before in the study of FFTs. Namely, it is a two dimensional recurrence so to calculate $M(k_1, k_2)$ we need all the values $M(l_1, l_2)$ with $l_1 \le k_1$ and $l_2 \le k_2$.

For comparison we note that the multiplication count for the row-column algorithm can be calculated as

$$M_{RC}(k_1,k_2) = 2^{k_1}M_{RC}(k_2) + 2^{k_2}M_{RC}(k_1)$$

which directly uses the multiplication count for the one dimensional row-column DFT and for the radix 2 vector algorithm we have

$$M_{vector}(k_1,k_2) = 4M_{vector}(k_1-1,k_2-1) + 3\cdot 2^{k_1+k_2-2}$$

so in order to be able to calculate $M_{vector}(k_1, k_2)$ we only need to calculate $M_{vector}(k_1 - l, k_2 - l)$ for l from 1 to $\min(k_1, k_2)$.

To analyze the system (11) and (12) further we note that we can satisfy (7) – (9) by setting

(13a) $$M(k_1,k_2) = (k_1+k_2)2^{k_1+k_2-2} = N\lg(N)/4$$

(13b) $$Mo(k_1,k_2) = M(k_1,k_2) + 2^{k_1+k_2-1}$$

(13c) $$Moo(k_1,k_2) = Mo(k_1,k_2) + 2^{k_1+k_2-1} = M(k_1,k_2) + 2^{k_1+k_2}$$

and hence (13a) is a solution to (11) with the boundary condition $M(k,0) = M(0,k) = k2^{k-2}$. This solution takes care of the term $2^{k_1-1}2^{k_2-1}$ and half of the boundary contribution in (12). In essense, it gives the multiplication count we would have if we had as an efficient algorithm for one dimensional DFTs as the one we are proposing for two dimensional DFTs. Since this is not the case we must write $M = M_1 + M_2$ where M_1 corresponds to (13a) and M_2 represents the extra multiplications coming from the one dimensional DFTs. We see that

(14) $$M_2(k_1,k_2) = M_2(k_1-1,k_2) + M_2(k_1,k_2-1)$$

with

(15) $$M_2(k,0) = M_2(0,k) = k2^{k-2} = N\lg(N)/4.$$

Clearly, $M_2(k_1, k_2)$ is less than $M_1(k_1, k_2)$.

It is possible to write down an explicit formula for $M_2(k_1, k_2)$ by counting the paths from (k_1, k_2) to a boundary point, let us say of the form $(i, 0)$, where each step consists of decreasing one coordinate by one. The second to last point reached must be $(i, 1)$, since the other possibility entails reaching the boundary earlier, at $(i + 1,0)$. The length of the path from (k_1, k_2) to $(i, 1)$ is $k_1 + k_2 - i - 1$ and the second coordinate is decreased $k_2 - 1$ times so the number of paths is $\binom{k_1+k_2-i-1}{k_2-1}$ and each such path contributes $M_2(i, 0) = i2^{i-2}$ to the value of $M_2(k_1, k_2)$. When we also consider paths that end at points of the form $(0, i)$ and sum we get

$$M_2(k_1,k_2) = \frac{1}{4}\sum_{i=1}^{k_1} i2^i \binom{k_1+k_2-i-1}{k_2-1} + \frac{1}{4}\sum_{i=1}^{k_2} i2^i \binom{k_1+k_2-i-1}{k_1-1}.$$

However, there does not exist a simple formula for $M_2(k_1, k_2)$. We can study the generating function for it, which is the function $F(x, y)$ defined as

(16) $$F(x,y) = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} M_2(k_1,k_2)x^{k_1}y^{k_2}.$$

By multiplying both sides of (14) with $x^{k_1}y^{k_2}$ and summing over all values of k_1 and k_2 for which it holds, namely $k_1, k_2 \ge 1$, we get that

$$\sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty} M_2(k_1,k_2)x^{k_1}y^{k_2} = \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty} M_2(k_1-1,k_2)x^{k_1}y^{k_2} + \sum_{k_1=1}^{\infty}\sum_{k_2=1}^{\infty} M_2(k_1,k_2-1)x^{k_1}y^{k_2}.$$

We rewrite each sum using (16) and the fact that $M_2(0,0) = 0$ and get

$$F(x,y) - \sum_{k_1=1}^{\infty} M_2(k_1,0)x^{k_1} - \sum_{k_2=1}^{\infty} M_2(0,k_2)y^{k_2} = x\left(F(x,y) - \sum_{k_1=1}^{\infty} M_2(k_1,0)x^{k_1}\right) + y\left(F(x,y) - \sum_{k_2=1}^{\infty} M_2(0,k_2)y^{k_2}\right).$$

Since $M_2(k, 0) = M_2(0, k) = k2^{k-2}$ we get that

$$\sum_{k_1=1}^{\infty} M_2(k_1,0)x^{k_1} = \frac{1}{4}\sum_{k_1=1}^{\infty} k_1(2x)^{k_1} = \frac{x}{2(1-2x)^2}.$$

Using the analogous formula for the terms involving $M_2(0,k_2)$ and solving for $F(x,y)$ gives

$$F(x,y) = \frac{1}{2(1-x-y)}\left(\frac{x(1-x)}{(1-2x)^2} + \frac{y(1-y)}{(1-2y)^2}\right). \tag{17}$$

We can use a method from [9] and [10] which is called the diagonal method in the theory of multivariate generating functions, e.g. [11], to derive an expression for the generating function for the diagonal terms $M_2(k,k)$ which correspond to the most interesting case, the square matrix when $k_1 = k_2 = k$. This generating function is

$$G_2(x) = \sum_{k=0}^{\infty} M_2(k,k)x^k.$$

The first step is to make a substitution in (16) to get

$$F\left(\frac{x}{t},t\right) = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} M_2(k_1,k_2)\left(\frac{x}{t}\right)^{k_1} t^{k_2}.$$

We re-index this sum by defining $l = k_2 - k_1$ and use the convention that $M_2(k_1,k_2) = 0$ if either index is negative. Then

$$F\left(\frac{x}{t},t\right) = \sum_{l=-\infty}^{\infty}\left(\sum_{k_1=-\infty}^{\infty} M_2(k_1,k_1+l)x^{k_1}\right)t^l.$$

We see that $G_2(x)$ appears as the inner sum in this expression when $l = 0$. By using the residue theorem from complex analysis we get from this that

$$G_2(x) = \frac{x}{(1-4x)^{\frac{3}{2}}} = x\sum_{k=0}^{\infty}\binom{-3/2}{k}(-4x)^k$$
$$= \sum_{k=1}^{\infty}\binom{-3/2}{k-1}(-4)^{k-1}x^k$$

where we have used the generalized binomial theorem.[1] If we compare this with the definition of $G_2(x)$ we see that

$$M_2(k,k) = \binom{-3/2}{k-1}(-4)^{k-1}$$

and if we simplify this using the definition of the binomial coefficient we get

$$M_2(k,k) = \frac{k}{2}\binom{2k}{k}. \tag{18}$$

In total we have

$$\begin{aligned} M(k,k) &= \frac{1}{2}k4^k + \frac{k}{2}\binom{2k}{k} \\ &\approx \frac{1}{2}k4^k + \frac{1}{2\sqrt{\pi}}k^{\frac{1}{2}}4^k \\ &\quad - \frac{1}{16\sqrt{\pi}}k^{-\frac{1}{2}}4^k + O\left(k^{-\frac{3}{2}}4^k\right) \\ &= \frac{1}{4}N\lg(N) + \frac{1}{\sqrt{8\pi}}N\sqrt{\lg(N)} \\ &\quad - \frac{1}{8\sqrt{2\pi}}\frac{N}{\sqrt{\lg(N)}} + O\left(\frac{N}{(\lg(N))^{\frac{3}{2}}}\right). \end{aligned} \tag{19}$$

where $N = 4^k$ is the total number of entries in the matrix.

Comparing this with the corresponding formula for the row-column algorithm,

$$M_{RC}(k,k) = k4^k = \frac{1}{2}N\lg(N) \tag{20}$$

we see that the dominant term, $N\lg(N)$, is reduced by a factor of 2 but there are new terms of order $N\sqrt{\lg(N)}$ and less. Numerical comparison of these counts can be seen in *Table 2*.

TABLE 2

Number of complex multiplications for the 2D radix 2 versions of the diagonal method and of the row—column method for a matrix of size $2^k \times 2^k$.

k	4	6	8	10	12
$M(k,k)$	652	15,060	313,624	6,166,660	116,888,232
$M_{RC}(k,k)$	1024	24,576	524,288	10,485,760	201,326,592
Ratio	63.7%	61.3%	59.8%	58.8%	58.1%

B. *Real Multiplications*

We now turn to the problem of counting real multiplications in the algorithms considers. We say that multiplication by a general complex number costs three real multiplications, multiplications by powers of ω_8 which are not powers of 4 costs two real multiplications and multiplication by powers of ω_4 costs no real multiplications. Then the number of real additions used for these complex multiplications is the same as the number of real multiplications.

The number of extra additions, i.e. those not associated with complex multiplications is the same for each of the algorithms considered,

$$A^{real}(k,k) = 4k2^{2k} = 2N\lg(N)$$

and the total operation count is $2M^{real} + A^{real}$.

For the row-column radix 2 algorithm the real multiplication count is as follows

$$\begin{aligned} M_{RC}^{real}(k,k) &= 3k2^{2k} - 10\cdot 2^{2k} + 16\cdot 2^k \\ &= \frac{3}{2}N\lg(N) - 10N + 16\sqrt{N} \end{aligned}$$

and for the vector radix 2 algorithm it is exactly one quarter less, or

[1]To explain this in more detail we note from (15) that $F(x,y)$ is a rational function which is defined (at least) in a neighbourhood of (0,0). Thus if x is small enough then $F(x/t,t)$ is absolutely convergent in some annulus around 0. In fact this annulus encloses exactly those poles of $F(x/t,t)$ which tend to 0 as x tends to 0. Therefore, the residue theorem gives that

$$G_2(x) = \sum_{\alpha}\operatorname{Res}\left(\frac{F(x/t,t)}{t},\alpha\right)$$

where the sum is over those poles α of $F(x/t,t)/t$ which tend to 0 as x tends to 0. Calculating these residues gives the stated form of $G_2(x)$.

$$M_{vector}^{real}(k,k) = \frac{9}{4}k2^{2k} - \frac{30}{4}2^{2k} + 12 \cdot 2^k$$
$$= \frac{9}{8}N\lg(N) - \frac{30}{4}N + 12\sqrt{N}.$$

Unfortunately, there does not exist such a simple formula for the real multiplication count for the diagonal radix 2 algorithm. However, it is possible to derive a good asymptotic formula. We use a recurrence relation for $M_{diag}^{real}(k_1,k_2)$ which is derived in a similar manner to the system (12) and (13). The relation is

$$M_{diag}^{real}(k_1,k_2) = M_{diag}^{real}(k_1-1,k_2) + M_{diag}^{real}(k_1,k_2-1)$$
$$+ 3 \cdot 2^{k_1+k_2-2} - 8 \cdot 2^{\min(k_1,k_2)-1}$$

which holds when $k_1,k_2 \geq 3$. The boundary conditions are

$$M_{diag}^{real}(k,l) = M_{diag}^{real}(l,k) = 2^l(3k2^{k-1} - 5\cdot 2^k + 8)$$

if $0 \leq l \leq 2$. We then introduce the generating function for $M_{diag}^{real}(k_1,k_2)$ given by the expression

$$F(x,y) = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} M_{diag}^{real}(k_1,k_2)x^{k_1}y^{k_2}.$$

With the recurrence relation we can find a closed form expression for $F(x,y)$. By then using the aforementioned diagonal method, we can extract an expression for the generating function for the diagonal terms $M_{diag}^{real}(k,k)$. This turns out to be

$$G(x) = \sum_{k=0}^{\infty} M_{diag}^{real}(k,k)x^k = \frac{16x^3(2x+1)\left(\sqrt{1-4x}+2\right)}{(1-4x)^2(1-2x)}.$$

We write $G(x)$ as

$$G(x) = \frac{3}{2}(1-4x)^{-2} + \frac{3}{4}(1-4x)^{-\frac{3}{2}} - \frac{13}{2}(1-4x)^{-1}$$
$$- \frac{13}{4}(1-4x)^{-\frac{1}{2}} + O(1)$$

and from this we can determine the asymptotics of $M_{diag}^{real}(k,k)$ which are

$$M_{diag}^{real}(k,k) \approx \frac{3}{2}k4^k + \frac{3}{2\sqrt{\pi}}k^{\frac{1}{2}}4^k - 5\cdot 4^k - \frac{43}{16\sqrt{\pi}}k^{-\frac{1}{2}}4^k$$
$$+ O\left(k^{-\frac{3}{2}}4^k\right)$$
$$\approx \frac{3}{4}N\lg(N) + \frac{3}{2\sqrt{2\pi}}N\sqrt{\lg(N)} - 5N - \frac{43}{8\sqrt{2\pi}}\frac{N}{\sqrt{\lg(N)}}$$
$$+ O\left(\frac{N}{(\lg(N))^{\frac{3}{2}}}\right).$$

These asymptotics are very accurate, apart from $M_{diag}^{real}(3,3) = 48$ and $M_{diag}^{real}(4,4) = 544$, the error is less than 1.2%. Further terms of this asymptotic expansion can be calculated but they do not yield greater accuracy for the lowest values of k.

For the row-column split radix algorithm the real multiplication count is as follows

$$M_{RC,SR}^{real}(k,k) = 2k2^{2k} - 6\cdot 2^{2k} + 8\cdot 2^k$$
$$= N\lg(N) - 6N + 8\sqrt{N}$$

and for the vector split radix algorithm it is

$$M_{vector,SR}^{real}(k,k) = \frac{9}{7}k4^k - \frac{183}{49}4^k - \frac{16}{245}(-3)^k + \frac{24}{5}2^k$$
$$= \frac{9}{14}N\lg(N) - \frac{183}{49}N$$
$$- \frac{16}{245}(-1)^{\frac{\lg(N)}{2}}N^{\frac{\lg(3)}{2}} + \frac{24}{5}\sqrt{N}.$$

There does not exist any simple formula for the real multiplication count for the diagonal split radix algorithm, as was the case for the radix 2 version. We therefore again seek to derive a good asymptotic formula. The recurrence relation for $M_{diag,SR}^{real}(k_1,k_2)$ is

$$M_{diag,SR}^{real}(k_1,k_2) = M_{diag,SR}^{real}(k_1-1,k_2)$$
$$+ M_{diag,SR}^{real}(k_1,k_2-1)$$
$$- M_{diag,SR}^{real}(k_1-1,k_2-1)$$
$$+ 4M_{diag,SR}^{real}(k_1-2,k_2-2) + 3\cdot 2^{k_1+k_2-2}$$
$$- 8\cdot 2^{\min(k_1,k_2)-1}$$

which holds when $k_1,k_2 \geq 3$. The boundary conditions are

$$M_{diag,SR}^{real}(k,l) = M_{diag,SR}^{real}(l,k) = 2^l(k2^k - 3\cdot 2^k + 4)$$

if $0 \leq l \leq 2$.

As before we can extract an expression for the generating function for the diagonal terms $M_{diag,SR}^{real}(k,k)$. This turns out to be

$$G_{SR}(x) = \sum_{k=0}^{\infty} M_{diag,SR}^{real}(k,k)x^k$$
$$= \frac{16x^3\left(\sqrt{1-4x} + 2\sqrt{(1-x)(1+3x+4x^2)}\right)}{(1-2x)\sqrt{(1-x)(1+3x+4x^2)}(1-4x)^2}.$$

We write $G_{SR}(x)$ as

$$G_{SR}(x) = (1-4x)^{-2} + \frac{1}{\sqrt{6}}(1-4x)^{-\frac{3}{2}} - 4(1-4x)^{-1}$$
$$- \frac{185}{48\sqrt{6}}(1-4x)^{-\frac{1}{2}} + O(1)$$

and from this we can determine the asymptotics of $M_{diag,SR}^{real}(k,k)$ which are

$$M_{diag,SR}^{real}(k,k) \approx k4^k + \sqrt{\frac{2}{3\pi}}k^{\frac{1}{2}}4^k - 3\cdot 4^k - \frac{149}{48\sqrt{6\pi}}k^{-\frac{1}{2}}4^k$$
$$+ O\left(k^{-\frac{3}{2}}4^k\right)$$
$$\approx \frac{1}{2}N\lg(N) + \frac{1}{\sqrt{3\pi}}N\sqrt{\lg(N)} - 3N - \frac{149}{48\sqrt{3\pi}}\frac{N}{\sqrt{\lg(N)}}$$
$$+ O\left(\frac{N}{(\lg(N))^{\frac{3}{2}}}\right).$$

These asymptotics are very accurate, apart from $M_{diag,SR}^{real}(3,3) = 48$ and $M_{diag,SR}^{real}(4,4) = 432$, the error is less than 1%. Again, further terms do not yield greater accuracy for the lowest values of k.

We turn now to radix 4. Here we will work with data of size $2^{2k_1} \times 2^{2k_2}$ and again focus on the square case, when $k_1 = k_2$. For the row-column radix 4 algorithm the real multiplication count is as follows

$$M_{RC,R4}^{real}(2k,2k) = \frac{9}{2}k2^{4k} - \frac{43}{6}2^{4k} + \frac{32}{3}2^{2k}$$
$$= \frac{9}{8}N\lg(N) - \frac{43}{6}N + \frac{32}{3}\sqrt{N}$$

and for the vector radix 4 algorithm it is

$$M_{vector,R4}^{real}(2k,2k) = \frac{45}{16}k2^{4k} - \frac{69}{16}2^{4k} + 6\cdot 2^{2k}$$
$$= \frac{45}{64}N\lg(N) - \frac{69}{16}N + 6\cdot\sqrt{N}.$$

Yet again there does not exist any simple formula for the real multiplication count for the diagonal radix 4 algorithm but we can derive a good asymptotic formula. The recurrence relation for $M_{diag,R4}^{real}(2k_1,2k_2)$ is

$$\begin{aligned} M_{diag,R4}^{real}(2k_1,2k_2) &= M_{diag,R4}^{real}(2k_1-2,2k_2) \\ &+ M_{diag,R4}^{real}(2k_1,2k_2-2) \\ &+ 8M_{diag,R4}^{real}(2k_1-2,2k_2-2) \\ &+ E(2k_1,2k_2) \end{aligned}$$

which holds when $k_1,k_2 \geq 1$. The extra multiplications $E(2k_1,2k_2)$ equal $\frac{27}{16}2^{2k_1+2k_2} - 12\cdot 2^{2\min(k_1,k_2)}$ if $k_1,k_2 \geq 1$ and $k_1 \neq k_2$; equal $\frac{27}{16}2^{4k} - 10\cdot 2^{2k}$ if $k_1 = k_2 = k \geq 2$; and 0 otherwise.

The boundary conditions are

$$M_{diag,R4}^{real}(2k,0) = M_{diag,R4}^{real}(0,2k) = \frac{9}{4}k2^{2k} - \frac{43}{12}2^{2k} + \frac{16}{3}$$

As before we can extract an expression for the generating function for the diagonal terms $M_{diag,R4}^{real}(2k,2k)$. This turns out to be

$$G_{R4}(x) = \sum_{k=0}^{\infty} M_{diag,R4}^{real}(2k,2k)x^k$$
$$= \frac{16\left((7-4x)\sqrt{1-16x} + (20+112x)\sqrt{1-4x}\right)}{(1-16x)^2(1-4x)^{\frac{3}{2}}}.$$

We write $G_{R4}(x)$ as

$$G_{SR}(x) = \frac{9}{4}(1-16x)^{-2} + \frac{3\sqrt{3}}{8}(1-16x)^{-\frac{3}{2}}$$
$$-\frac{35}{6}(1-16x)^{-1} - \frac{133\sqrt{3}}{144}(1-16x)^{-\frac{1}{2}}$$
$$+ O(1)$$

and from this we can determine the asymptotics of $M_{diag,R4}^{real}(2k,2k)$ which are

$$M_{diag,SR}^{real}(2k,2k) \approx \frac{9}{4}k2^{4k} + \frac{3}{4}\sqrt{\frac{3}{\pi}}k^{\frac{1}{2}}2^{4k} - \frac{43}{12}\cdot 4^i$$
$$-\frac{185}{288}\sqrt{\frac{3}{\pi}}k^{-\frac{1}{2}}2^{4k} + O\left(k^{-\frac{3}{2}}2^{4k}\right)$$
$$\approx \frac{9}{16}N\lg(N) + \frac{3}{8}\sqrt{\frac{3}{\pi}}N\sqrt{\lg(N)} - \frac{43}{12}N - \frac{185}{144}\sqrt{\frac{3}{\pi}}\frac{N}{\sqrt{\lg(N)}}$$
$$+ O\left(\frac{N}{(\lg(N))^{\frac{3}{2}}}\right).$$

These asymptotics are very accurate, apart from $M_{diag,R4}^{real}(4,4) = 432$, the error is less than 0.5%. Again, further terms do not yield greater accuracy for the lowest values of $2k$.

C. *Numerical results on operation count*

The following table presents numerical results of the multiplication count. These numbers are all extracted from a C implementation we have written for these algorithms. The theoretical results of this section match with these numbers. Note that number of multiplications is smallest for the Nussbaumer and Quandalle algorithm followed closely by the Diagonal algorithm. Furthermore, the difference between these two multiplication counts is in each case far smaller than the number of operations in the extra permutation steps which the Nussbaumer and Quandalle algorithm uses. Thus the Diagonal algorithm may be expected to perform the best on modern computer architectures. This is indeed what we have found in practice with the caveat that none of our implementations have been optimized for speed. For that reason we do not present running time data. Nevertheless, we feel that our results show that the diagonal algorithm warrants strong consideration as a simple, fast and practical multidimensional FFT algorithm.

TABLE 3
Number of real multiplications for a 2D matrix of size $2^k \times 2^k$ divided by the number of output points ($2 \times 2^k \times 2^k$). Here R2 denotes Radix 2, R4 denotes Radix 4, SR denotes Split Radix, and NQ is the polynomial method of Nussbaumer and Quandalle. According to [3], the numbers for the method of Bernardini are the same as for the NQ method.

k	4	6	8	10	12
R2 Row/Col	1.50	4.13	7.03	10.01	13.00
R2 Vector	1.13	3.09	5.27	7.51	9.75
R2 Diagonal	1.06	2.70	4.39	6.07	7.72
R2 NQ	1.03	2.50	4.00	5.50	7.00
R4 Row/Col	1.25	3.25	5.44	7.67	9.92
R4 Vector	0.84	2.11	3.48	4.88	6.28
R4 Diagonal	0.84	2.03	3.26	4.49	5.71
R4 NQ	0.84	1.95	3.08	4.20	5.33
SR Row/Col	1.25	3.06	5.02	7.00	9.00
SR Vector	0.84	2.02	3.28	4.56	5.85
SR Diagonal	0.84	1.91	3.01	4.10	5.18
SR NQ	0.84	1.83	2.83	3.83	4.83

D. *Higher dimensions*

Going to m dimensions we have from (2), (5) and (6), similarly as before, that

(21a) $$\begin{aligned} M(k_1,\dots,k_m) &= M(k_1-1,k_2,\dots,k_m) \\ &+ Mo_1(k_1-1,k_2,\dots,k_m), \end{aligned}$$

(21b) $$\begin{aligned} &Mo_1 \dots o_i(k_1-1,\dots,k_i-1,k_{i+1},\dots,k_m) \\ &= Mo_1 \dots o_i(k_1-1,\dots,k_{i+1}-1,k_{i+2},\dots,k_m) \\ &+Mo_1 \dots o_{i+1}(k_1-1,\dots,k_{i+1}-1,k_{i+2},\dots,k_m) \end{aligned}$$

(21c) $$\begin{aligned} &Mo_1 \dots o_m(k_1-1,\dots,k_m-1) \\ &= M(k_1-1,\dots,k_m-1) \\ &+ 2^{k_1-1}\dots 2^{k_m-1} \end{aligned}$$

where (21b) holds for $i = 1, \dots, m-1$. We can, in turn, eliminate the terms $Mo_1 \dots o_i$ for $i = 1, \dots, m$ from these equationsto obtain an inhomogeneous recurrence relation for M. For $m = 3$ we thus e.g. get that

$$\begin{aligned} M(k_1,k_2,k_3) = {} & M(k_1-1,k_2,k_3) \\ & + M(k_1,k_2-1,k_3) \\ & + M(k_1,k_2,k_3-1) \\ & - M(k_1-1,k_2-1,k_3) \\ & - M(k_1-1,k_2,k_3-1) \\ & - M(k_1,k_2-1,k_3-1) \\ & + 2M(k_1-1,k_2-1,k_3-1) \\ & + 2^{k_1+k_2+k_3-3}. \end{aligned} \tag{22}$$

Further it is readily seen that we can satisfy the system (21) by setting

$$\begin{aligned} M(k_1,\dots,k_m) &= (k_1+\cdots+k_m)2^{k_1+\cdots+k_m}/m \\ &= N\lg(N)/(2m) \end{aligned} \tag{23a}$$

and

$$\begin{aligned} & Mo_1 \dots o_i(k_1,\dots,k_m) \\ & = Mo_1 \dots o_{i-1}(k_1,\dots,k_m) + 2^{k_1+\cdots+k_m}/m \\ & = M(k_1,\dots,k_m) + i2^{k_1+\cdots+k_m}/m \end{aligned} \tag{23b}$$

for $i = 1, \dots, m$. Thus (23a) will be a solution to the corresponding recurrence relation in M satisfying the boundary condition

$$M(0,k_2,\dots,k_m) = (k_2+\cdots+k_m)2^{k_2+\cdots+k_m}$$

and similar boundary conditions when some other $k_i = 0$, for $i = 2, \dots, m$.

However, the solution we are seeking must satisfy the condition that $M(0,k_2,\dots,k_m)$ is the multiplication count corresponding to a transform of dimension $m-1$. Since the corresponding algorithm is less efficient than that of the transformation of dimension m we must take care of the difference in multiplications by expressing the solution as

$$M(k_1,\dots,k_m) = M_1(k_1,\dots,k_m) + M_2(k_1,\dots,k_m) \tag{24}$$

where M_1 is given by (23a) and M_2 is the solution of the corresponding homogeneous recurrence relation in M with boundary conditions that take care of the difference in multiplications.

It is possible to write down the generating function for M_2 but the diagonal method is not applicable and therefore we cannot find a generating function for the diagonal terms. However, there is evidence that the asymptotic order of M_2 is less than $O(N\lg(N))$ and we tentatively conjecture that it is $O\left(N(\lg N)^{\frac{1}{2}}\right)$. Then asymptotically we have that the multiplication count for the m dimensional diagonal method is one m'th of the corresponding operation count for one dimensional data of the same size.

Let us briefly outline how further work in these asymptotic studies could proceed. It is possible to derive asympotic formulas directly from multivariate generating functions. This area of research has been quite active in the last decade or so, see e.g. the recent book [11]. To understand how the theory applies let us go back to two dimensions and the simplest example, namely equation (17). We can write $F(x,y)$ as the sum of two rational functions, one of which is

$$\tilde{F}(x,y) = \frac{x(1-x)}{2(1-x-y)(1-2x)^2} = \sum_{k_1=0}^{\infty}\sum_{k_2=0}^{\infty} a_{k_1,k_2}x^{k_1}y^{k_2}.$$

It turns out that the asymptotics of the coefficients a_{k_1,k_2} depend on the direction of the vector (k_1,k_2). If (k_1,k_2) goes to infinity along a line through the origin with slope less than 1 then the asymptotic form of the coefficients is dominated by a point on the line $x = 1/2$ where the second factor in the denominator is 0. Such a point is called a smooth point in [11]. But if the slope of the line which (k_1,k_2) goes to infinity along is greater than 1 then the asymptotic form of the coefficients is dominated by the point $(1/2,1/2)$ where both factors in the denominator are 0. Such a point is called a multiple point in [11]. However, the line we are most interested in, when $k_1 = k_2$ falls exactly between these two cases and this border case is not studied in as great a detail in [11] as the other two cases. Therefore there is quite a bit of manual labor to be done to apply this new theory in our setting.

References

[1] P. Duhamel and M. Vetterli, "Fast Fourier-transforms: a tutorial review and a state of the art," *Signal Process.*, vol. 19, no. 4, pp. 259-299, Apr 1990.

[2] H. J. Nussbaumer and P. Quandalle, "Fast computation of discrete Fourier-transforms using polynomial transforms," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 27, no. 2, pp. 169-181, 1979.

[3] R. Bernardini, "A new multidimensional FFT based on one-dimensional decompositions," *IEEE Trans. Circ. and Syst. II*, vol. 47, no. 10, pp. 1123-1126, Oct 2000.

[4] R. E. Blahut, *Fast algorithms for signal processing*. Cambridge, UK: Cam. Uni. Press, 2010.

[5] Z. D. Chen and L. J. Zhang, "Vector coding algorithms for multidimensional discrete Fourier transform," *J Comput Appl Math*, vol. 212, no. 1, pp. 63-74, Jan 2008.

[6] M. Frigo and S. G. Johnson, "The design and implementation of FFTW3," *Proc. IEEE*, vol. 93, no. 2, pp. 216-231, Feb 2005.

[7] I. Kamar and Y. Elcherif, "Conjugate pair fast Fourier-transform," *Electron. Lett.*, vol. 25, no. 5, pp. 324-325, Mar 1989.

[8] R. A. Gopinath, "Comment on: 'Conjugate pair fast Fourier transform'," *Electron. Lett.*, vol. 25, no. 16, p. 1084, Aug 1989.

[9] H. Furstenberg, "Algebraic functions over finite fields," *J. Algebra*, vol. 7, pp. 271--277, 1967.

[10] M. L. J. Hautus and D. A. Klarner, "The diagonal of a double power series," *Duke Math. J.*, vol. 38, pp. 229--235, 1971.

[11] R. Pemantle and M. C. Wilson, *Analytic combinatorics in several variables*. Cambridge, UK: Cam. Uni. Press., 2013.