Fast Fourier Transforms

Collection Editor:
C. Sidney Burrus

Fast Fourier Transforms

Collection Editor:

C. Sidney Burrus

Authors:

C. Sidney Burrus
Matteo Frigo
Steven G. Johnson
Markus Pueschel
Ivan Selesnick

Online:

< http://cnx.org/content/col10550/1.22/ >

C O N N E X I O N S

Rice University, Houston, Texas

Table of Contents

Chapter 1

Preface: Fast Fourier Transforms[1]

This book focuses on the discrete Fourier transform (DFT), discrete convolution, and, particularly, the fast algorithms to calculate them. These topics have been at the center of digital signal processing since its beginning, and new results in hardware, theory and applications continue to keep them important and exciting.

As far as we can tell, Gauss was the first to propose the techniques that we now call the fast Fourier transform (FFT) for calculating the coefficients in a trigonometric expansion of an asteroid's orbit in 1805 [174]. However, it was the seminal paper by Cooley and Tukey [88] in 1965 that caught the attention of the science and engineering community and, in a way, founded the discipline of digital signal processing (DSP).

The impact of the Cooley-Tukey FFT was enormous. Problems could be solved quickly that were not even considered a few years earlier. A flurry of research expanded the theory and developed excellent practical programs as well as opening new applications [94]. In 1976, Winograd published a short paper [403] that set a second flurry of research in motion [86]. This was another type of algorithm that expanded the data lengths that could be transformed efficiently and reduced the number of multiplications required. The ground work for this algorithm had be set earlier by Good [148] and by Rader [308]. In 1997 Frigo and Johnson developed a program they called the FFTW (fastest Fourier transform in the west) [130], [135] which is a composite of many of ideas in other algorithms as well as new results to give a robust, very fast system for general data lengths on a variety of computer and DSP architectures. This work won the 1999 Wilkinson Prize for Numerical Software.

It is hard to overemphasis the importance of the DFT, convolution, and fast algorithms. With a history that goes back to Gauss [174] and a compilation of references on these topics that in 1995 resulted in over 2400 entries [362], the FFT may be the most important numerical algorithm in science, engineering, and applied mathematics. New theoretical results still are appearing, advances in computers and hardware continually restate the basic questions, and new applications open new areas for research. It is hoped that this book will provide the

[1]This content is available online at <http://cnx.org/content/m16324/1.10/>.

background, references, programs and incentive to encourage further research and results in this area as well as provide tools for practical applications.

Studying the FFT is not only valuable in understanding a powerful tool, it is also a prototype or example of how algorithms can be made efficient and how a theory can be developed to define optimality. The history of this development also gives insight into the process of research where timing and serendipity play interesting roles.

Much of the material contained in this book has been collected over 40 years of teaching and research in DSP, therefore, it is difficult to attribute just where it all came from. Some comes from my earlier FFT book [59] which was sponsored by Texas Instruments and some from the FFT chapter in [217]. Certainly the interaction with people like Jim Cooley and Charlie Rader was central but the work with graduate students and undergraduates was probably the most formative. I would particularly like to acknowledge Ramesh Agarwal, Howard Johnson, Mike Heideman, Henrik Sorensen, Doug Jones, Ivan Selesnick, Haitao Guo, and Gary Sitton. Interaction with my colleagues, Tom Parks, Hans Schuessler, Al Oppenheim, and Sanjit Mitra has been essential over many years. Support has come from the NSF, Texas Instruments, and the wonderful teaching and research environment at Rice University and in the IEEE Signal Processing Society.

Several chapters or sections are written by authors who have extensive experience and depth working on the particular topics. Ivan Selesnick had written several papers on the design of short FFTs to be used in the prime factor algorithm (PFA) FFT and on automatic design of these short FFTs. Markus P*ü*schel has developed a theoretical framework for "Algebraic Signal Processing" which allows a structured generation of FFT programs and a system called "Spiral" for automatically generating algorithms specifically for an architicture. Steven Johnson along with his colleague Matteo Frigo created, developed, and now maintains the powerful FFTW system: the Fastest Fourier Transform in the West. I sincerely thank these authors for their significant contributions.

I would also like to thank Prentice Hall, Inc. who returned the copyright on The DFT as Convolution or Filtering (Chapter 5) of **Advanced Topics in Signal Processing** [49] around which some of this book is built. The content of this book is in the Connexions (http://cnx.org/content/col10550/) repository and, therefore, is available for on-line use, **pdf** down loading, or purchase as a printed, bound physical book. I certainly want to thank Daniel Williamson, Amy Kavalewitz, and the staff of Connexions for their invaluable help. Additional FFT material can be found in Connexions, particularly content by Doug Jones [205], Ivan Selesnick [205], and Howard Johnson, [205]. Note that this book and all the content in Connexions are copyrighted under the Creative Commons Attribution license (http://creativecommons.org/).

If readers find errors in any of the modules of this collection or have suggestions for improvements or additions, please email the author of the collection or module.

C. Sidney Burrus

Houston, Texas

October 20, 2008

Chapter 2

Introduction: Fast Fourier Transforms[1]

The development of fast algorithms usually consists of using special properties of the algorithm of interest to remove redundant or unnecessary operations of a direct implementation. Because of the periodicity, symmetries, and orthogonality of the basis functions and the special relationship with convolution, the discrete Fourier transform (DFT) has enormous capacity for improvement of its arithmetic efficiency.

There are four main approaches to formulating efficient DFT [50] algorithms. The first two break a DFT into multiple shorter ones. This is done in Multidimensional Index Mapping (Chapter 3) by using an index map and in Polynomial Description of Signals (Chapter 4) by polynomial reduction. The third is Factoring the Signal Processing Operators (Chapter 6) which factors the DFT operator (matrix) into sparse factors. The DFT as Convolution or Filtering (Chapter 5) develops a method which converts a prime-length DFT into cyclic convolution. Still another approach is interesting where, for certain cases, the evaluation of the DFT can be posed recursively as evaluating a DFT in terms of two half-length DFTs which are each in turn evaluated by a quarter-length DFT and so on.

The very important computational complexity theorems of Winograd are stated and briefly discussed in Winograd's Short DFT Algorithms (Chapter 7). The specific details and evaluations of the Cooley-Tukey FFT and Split-Radix FFT are given in The Cooley-Tukey Fast Fourier Transform Algorithm (Chapter 9), and PFA and WFTA are covered in The Prime Factor and Winograd Fourier Transform Algorithms (Chapter 10). A short discussion of high speed convolution is given in Convolution Algorithms (Chapter 13), both for its own importance, and its theoretical connection to the DFT. We also present the chirp, Goertzel, QFT, NTT, SR-FFT, Approx FFT, Autogen, and programs to implement some of these.

Ivan Selesnick gives a short introduction in Winograd's Short DFT Algorithms (Chapter 7) to using Winograd's techniques to give a highly structured development of short prime length FFTs and describes a program that will automaticlly write these programs. Markus Pueschel presents his "Algebraic Signal Processing" in DFT and FFT: An Algebraic View (Chapter 8)

[1] This content is available online at <http://cnx.org/content/m16325/1.10/>.

on describing the various FFT algorithms. And Steven Johnson describes the FFTW (Fastest Fourier Transform in the West) in Implementing FFTs in Practice (Chapter 11)

The organization of the book represents the various approaches to understanding the FFT and to obtaining efficient computer programs. It also shows the intimate relationship between theory and implementation that can be used to real advantage. The disparity in material devoted to the various approaches represent the tastes of this author, not any intrinsic differences in value.

A fairly long list of references is given but it is impossible to be truly complete. I have referenced the work that I have used and that I am aware of. The collection of computer programs is also somewhat idiosyncratic. They are in Matlab and Fortran because that is what I have used over the years. They also are written primarily for their educational value although some are quite efficient. There is excellent content in the Connexions book by Doug Jones [206].

Chapter 3

Multidimensional Index Mapping[1]

A powerful approach to the development of efficient algorithms is to break a large problem into multiple small ones. One method for doing this with both the DFT and convolution uses a linear change of index variables to map the original one-dimensional problem into a multi-dimensional problem. This approach provides a unified derivation of the Cooley-Tukey FFT, the prime factor algorithm (PFA) FFT, and the Winograd Fourier transform algorithm (WFTA) FFT. It can also be applied directly to convolution to break it down into multiple short convolutions that can be executed faster than a direct implementation. It is often easy to translate an algorithm using index mapping into an efficient program.

The basic definition of the discrete Fourier transform (DFT) is

$$C(k) = \sum_{n=0}^{N-1} x(n) \quad W_N^{nk} \tag{3.1}$$

where n, k, and N are integers, $j = \sqrt{-1}$, the basis functions are the N roots of unity,

$$W_N = e^{-j2\pi/N} \tag{3.2}$$

and $k = 0, 1, 2, \cdots, N-1$.

If the N values of the transform are calculated from the N values of the data, $x(n)$, it is easily seen that N^2 complex multiplications and approximately that same number of complex additions are required. One method for reducing this required arithmetic is to use an index mapping (a change of variables) to change the one-dimensional DFT into a two- or higher dimensional DFT. This is one of the ideas behind the very efficient Cooley-Tukey [89] and Winograd [404] algorithms. The purpose of index mapping is to change a large problem into several easier ones [46], [120]. This is sometimes called the “divide and conquer" approach [26] but a more accurate description would be “organize and share" which explains the process of redundancy removal or reduction.

[1]This content is available online at <http://cnx.org/content/m16326/1.12/>.

3.1 The Index Map

For a length-N sequence, the time index takes on the values

$$n = 0, 1, 2, ..., N-1 \tag{3.3}$$

When the length of the DFT is not prime, N can be factored as $N = N_1 N_2$ and two new independent variables can be defined over the ranges

$$n_1 = 0, 1, 2, ..., N_1 - 1 \tag{3.4}$$

$$n_2 = 0, 1, 2, ..., N_2 - 1 \tag{3.5}$$

A linear change of variables is defined which maps n_1 and n_2 to n and is expressed by

$$n = ((K_1 n_1 + K_2 n_2))_N \tag{3.6}$$

where K_i are integers and the notation $((x))_N$ denotes the integer residue of x modulo N[232]. This map defines a relation between all possible combinations of n_1 and n_2 in (3.4) and (3.5) and the values for n in (3.3). The question as to whether all of the n in (3.3) are represented, i.e., whether the map is one-to-one (unique), has been answered in [46] showing that certain integer K_i always exist such that the map in (3.6) is one-to-one. Two cases must be considered.

3.1.1 Case 1.

N_1 and N_2 are relatively prime, i.e., the greatest common divisor $(N_1, N_2) = 1$.

The integer map of (3.6) is one-to-one if and only if:

$$(K_1 = aN_2) \quad \text{and/or} \quad (K_2 = bN_1) \quad \text{and} \quad (K_1, N_1) = (K_2, N_2) = 1 \tag{3.7}$$

where a and b are integers.

3.1.2 Case 2.

N_1 and N_2 are not relatively prime, i.e., $(N_1, N_2) > 1$.

The integer map of (3.6) is one-to-one if and only if:

$$(K_1 = aN_2) \quad \text{and} \quad (K_2 \neq bN_1) \quad \text{and} \quad (a, N_1) = (K_2, N_2) = 1 \tag{3.8}$$

or

$$(K_1 \neq aN_2) \quad \text{and} \quad (K_2 = bN_1) \quad \text{and} \quad (K_1, N_1) = (b, N_2) = 1 \tag{3.9}$$

Reference [46] should be consulted for the details of these conditions and examples. Two classes of index maps are defined from these conditions.

3.1.3 Type-One Index Map:

The map of (3.6) is called a type-one map when integers a and b exist such that

$$K_1 = aN_2 \ \text{ and } \ K_2 = bN_1 \tag{3.10}$$

3.1.4 Type-Two Index Map:

The map of (3.6) is called a type-two map when when integers a and b exist such that

$$K_1 = aN_2 \ \text{ or } \ K_2 = bN_1, \quad \text{but not both.} \tag{3.11}$$

The type-one can be used **only** if the factors of N are relatively prime, but the type-two can be used whether they are relatively prime or not. Good [149], Thomas, and Winograd [404] all used the type-one map in their DFT algorithms. Cooley and Tukey [89] used the type-two in their algorithms, both for a fixed radix $\left(N = R^M\right)$ and a mixed radix [301].

The frequency index is defined by a map similar to (3.6) as

$$k = ((K_3 k_1 + K_4 k_2))_N \tag{3.12}$$

where the same conditions, (3.7) and (3.8), are used for determining the uniqueness of this map in terms of the integers K_3 and K_4.

Two-dimensional arrays for the input data and its DFT are defined using these index maps to give

$$\hat{x}\,(n_1, n_2) = x((K_1 n_1 + K_2 n_2))_N \tag{3.13}$$

$$\hat{X}\,(k_1, k_2) = X((K_3 k_1 + K_4 k_2))_N \tag{3.14}$$

In some of the following equations, the residue reduction notation will be omitted for clarity. These changes of variables applied to the definition of the DFT given in (3.1) give

$$C\,(k) = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x\,(n)\ W_N^{K_1 K_3 n_1 k_1}\ W_N^{K_1 K_4 n_1 k_2}\ W_N^{K_2 K_3 n_2 k_1}\ W_N^{K_2 K_4 n_2 k_2} \tag{3.15}$$

where all of the exponents are evaluated modulo N.

The amount of arithmetic required to calculate (3.15) is the same as in the direct calculation of (3.1). However, because of the special nature of the DFT, the integer constants K_i can be chosen in such a way that the calculations are "uncoupled" and the arithmetic is reduced. The requirements for this are

$$((K_1 K_4))_N = 0 \quad \text{and/or} \quad ((K_2 K_3))_N = 0 \tag{3.16}$$

When this condition and those for uniqueness in (3.6) are applied, it is found that the K_i may **always** be chosen such that one of the terms in (3.16) is zero. If the N_i are relatively prime, it is always possible to make **both** terms zero. If the N_i are not relatively prime, only one of the terms can be set to zero. When they are relatively prime, there is a choice, it is possible to either set one or both to zero. This in turn causes one or both of the center two W terms in (3.15) to become unity.

An example of the Cooley-Tukey radix-4 FFT for a length-16 DFT uses the type-two map with $K_1 = 4$, $K_2 = 1$, $K_3 = 1$, $K_4 = 4$ giving

$$n = 4n_1 + n_2 \tag{3.17}$$

$$k = k_1 + 4k_2 \tag{3.18}$$

The residue reduction in (3.6) is not needed here since n does not exceed N as n_1 and n_2 take on their values. Since, in this example, the factors of N have a common factor, only one of the conditions in (3.16) can hold and, therefore, (3.15) becomes

$$\hat{C}(k_1, k_2) = C(k) = \sum_{n_2=0}^{3} \sum_{n_1=0}^{3} x(n)\ W_4^{n_1 k_1}\ W_{16}^{n_2 k_1}\ W_4^{n_2 k_2} \tag{3.19}$$

Note the definition of W_N in (3.3) allows the simple form of $W_{16}^{K_1 K_3} = W_4$

This has the form of a two-dimensional DFT with an extra term W_{16}, called a "twiddle factor". The inner sum over n_1 represents four length-4 DFTs, the W_{16} term represents 16 complex multiplications, and the outer sum over n_2 represents another four length-4 DFTs. This choice of the K_i "uncouples" the calculations since the first sum over n_1 for $n_2 = 0$ calculates the DFT of the first row of the data array $\hat{x}(n_1, n_2)$, and those data values are never needed in the succeeding row calculations. The row calculations are independent, and examination of the outer sum shows that the column calculations are likewise independent. This is illustrated in Figure 3.1.

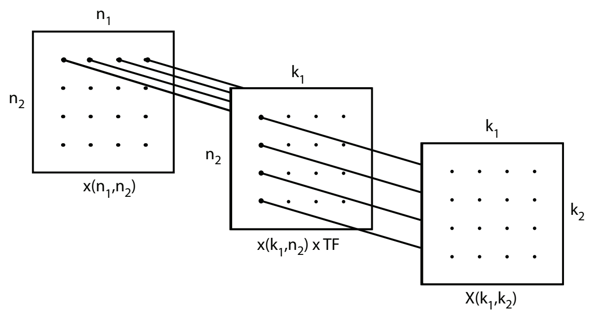

Figure 3.1: Uncoupling of the Row and Column Calculations (Rectangles are Data Arrays)

The left 4-by-4 array is the mapped input data, the center array has the rows transformed, and the right array is the DFT array. The row DFTs and the column DFTs are independent of each other. The twiddle factors (TF) which are the center W in (3.19), are the multiplications which take place on the center array of Figure 3.1.

This uncoupling feature reduces the amount of arithmetic required and allows the results of each row DFT to be written back over the input data locations, since that input row will not be needed again. This is called “in-place" calculation and it results in a large memory requirement savings.

An example of the type-two map used when the factors of N are relatively prime is given for $N = 15$ as

$$n = 5n_1 + n_2 \tag{3.20}$$

$$k = k_1 + 3k_2 \tag{3.21}$$

The residue reduction is again not explicitly needed. Although the factors 3 and 5 are relatively prime, use of the type-two map sets only one of the terms in (3.16) to zero. The DFT in (3.15) becomes

$$X = \sum_{n_2=0}^{4} \sum_{n_1=0}^{2} x \; W_3^{n_1 k_1} \; W_{15}^{n_2 k_1} \; W_5^{n_2 k_2} \tag{3.22}$$

which has the same form as (3.19), including the existence of the twiddle factors (TF). Here the inner sum is five length-3 DFTs, one for each value of k_1. This is illustrated in (3.2) where the rectangles are the 5 by 3 data arrays and the system is called a “mixed radix" FFT.