Henri J. Nussbaumer

Fast Fourier Transform and Convolution Algorithms

With 34 Figures

Springer-Verlag Berlin Heidelberg New York 1981

Dr Henri J. Nussbaumer

IBM Centre d'Etudes et Recherches
F-06610 La Gaude, Alpes-Maritimes, France

Series Editors

Professor King Sun Fu

School of Electrical Engineering, Purdue University
West Lafayette, IN 47907 USA

Professor Thomas S Huang

Department of Electrical Engineering and Coordinated Science Laboratory,
University of Illinois, Urbana IL 61801, USA

Professor Dr Manfred R. Schroeder

Drittes Physikalisches Institut, Universität Gottingen, Bürgerstraße 42–44,
D-3400 Gottingen, Fed Rep of Germany

ISBN 3-540-10159-4 Springer-Verlag Berlin Heidelberg New York
ISBN 0-387-10159-4 Springer-Verlag New York Heidelberg Berlin

Library of Congress Cataloging in Publication Data Nussbaumer Henri J 1931 - Fast Fourier transform and convolution algorithms (Springer series in information sciences, v 2) Bibliography p Includes index 1 Fourier transformations Data processing 2 Convolutions (Mathematics) Data processing 3 Digital filters (Mathematics) I Title II Series QA403 5 N87 515 7'23 80-18096

Offset printing and bookbinding Brühlsche Universitätsdruckerei, Giessen
2153/3130-543210

Preface

This book presents in a unified way the various fast algorithms that are used for the implementation of digital filters and the evaluation of discrete Fourier transforms.

The book consists of eight chapters The first two chapters are devoted to background information and to introductory material on number theory and polynomial algebra. This section is limited to the basic concepts as they apply to other parts of the book. Thus, we have restricted our discussion of number theory to congruences, primitive roots, quadratic residues, and to the properties of Mersenne and Fermat numbers. The section on polynomial algebra deals primarily with the divisibility and congruence properties of polynomials and with algebraic computational complexity.

The rest of the book is focused directly on fast digital filtering and discrete Fourier transform algorithms. We have attempted to present these techniques in a unified way by using polynomial algebra as extensively as possible. This objective has led us to reformulate many of the algorithms which are discussed in the book. It has been our experience that such a presentation serves to clarify the relationship between the algorithms and often provides clues to improved computation techniques.

Chapter 3 reviews the fast digital filtering algorithms, with emphasis on algebraic methods and on the evaluation of one-dimensional circular convolutions.

Chapters 4 and 5 present the fast Fourier transform and the Winograd Fourier transform algorithm.

We introduce in Chaps. 6 and 7 the concept polynomial transforms and we show that these transforms are an important tool for the understanding of the structure of multidimensional convolutions and discrete Fourier transforms and for the design of improved algorithms. In Chap. 8, we extend these concepts to the computation of one-dimensional convolutions by replacing finite fields of polynomials by finite fields of numbers This facilitates introduction of number theoretic transforms which are useful for the fast computation of convolutions via modular arithmetic.

Convolutions and discrete Fourier transforms have many uses in physics and it is our hope that this book will prompt some additional research in these areas and will help potential users to evaluate and apply these techniques. We also feel that some of the methods presented here are quite general and might someday find new unexpected applications.

Part of the material presented here has evolved from a graduate-level course taught at the University of Nice, France. I would like to express my thanks to Dr. T.A. Kriz from IBM FSD for kindly reviewing the manuscript and for making many useful suggestions. I am grateful to Mr. P. Bellot, IBM, C.E.R., La Gaude, France, for his advice concerning the introductory chapter on number theory and polynomial algebra, and to Dr. J. W. Cooley, from IBM Research, Yorktown Heights, for his comments on some of the work which led to this book. Thanks are also due to Dr. P. Quandalle who worked with me on polynomial transforms while preparing his doctorate degree and with whom I had many fruitful discussions. I am indebted to Mrs. C. De Backer for her aid in improving the English and to Mrs. C. Chevalier who prepared the manuscript.

La Gaude
November 1980

HENRI J. NUSSBAUMER

Contents

1. Introduction

1.1 Introductory Remarks

The practical applications of the digital convolution and of the discrete Fourier transform (DFT) have gained growing importance over the last few years. This is a direct consequence of the major role played by digital filtering and DFTs in digital signal processing and by the increasing use of digital signal processing techniques made possible by the rapidly declining cost of digital hardware. The motivation for developing fast convolution and DFT algorithms is strongly rooted in the fact that the direct computation of length-N convolutions and DFTs requires a number of operations proportional to N^2 which becomes rapidly excessive for long dimensions. This, in turn, implies an excessively large requirement for computer implementation of the methods.

Historically, the most important event in fast algorithm development has been the fast Fourier transform (FFT), introduced by Cooley and Tukey in 1965, which computes DFTs with a number of operations proportional to $N \log N$ and therefore reduces drastically the computational complexity for large transforms. Since convolutions can be computed by DFTs, the FFT algorithm can also be used to compute convolutions with a number of operations proportional to $N \log N$ and has therefore played a key role in digital signal processing ever since its introduction. More recently, many new fast convolution and DFT techniques fave been proposed to further decrease the computational complexity corresponding to these operations. The fast DFT algorithm introduced in 1976 by Winograd is perhaps the most important of these methods because it achieves a theoretical reduction of computational complexity over the FFT by a method which can be viewed as the converse of the FFT, since it computes a DFT as a convolution. Indeed, as we shall see in this book, the relationship between convolution and DFT has many facets and its implications go far beyond a mere algorithmic procedure.

Another important factor in the development of new algorithms was the recognition that convolutions and DFTs can be viewed as operations defined in finite rings and fields of integers and of polynomials. This new point of view has allowed both derivation of some lower computational complexity bounds and design of new and improved computation techniques such as those based on polynomial transforms and number theoretic transforms.

In addition to their practical implications, many convolution and DFT algorithms are also of theoretical significance because they lead to a better understanding of mathematical structures which may have many applications in areas other than convolution and DFT. It is likely, for instance, that polynomial transforms will appear as a very general tool for mapping multidimensional problems into one-dimensional problems.

The matter of comparing different algorithms which perform the same functions is pervasive throughout this book. In many cases, we have used the number of arithmetic operations required to execute an algorithm as a measure of the computational complexity. While there is some rough relationship between the overall complexity of an algorithm and its algebraic complexity, the practical value of a computation method depends upon a number of factors. Apart from the number of arithmetic operations, the efficiency of an algorithm is related to many parameters such as the number of data moves, the cost of ancillary operation, the overall structural complexity, the performance capabilities of the computer on which the algorithm is executed, and the skill of the programmer. Therefore, ranking different algorithms as a function of actual efficiency expressed in terms of computer execution times is a difficult art so that the comparisons based on the number of arithmetic operations must be weighted as a function of the particular implementation.

1.2 Notations

It is always difficult to avoid the proliferation of different symbols and subscripts when presenting the various DFT and convolution algorithms. We have adopted here some conventions in order to simplify the presentation. Discrete data sequences are usually represented by lower case letters such as x_n. We have not used the representation $\{x_n\}$ for data sequences, because this simplifies the notation and because the context information prevents confusion between the sequence and the n^{th} element of the sequence. Thus, in our representation, a discrete-time signal x_n is a sequence of the values of a continuous signal $x(t)$, sampled at times $t = nT$ and represented by a number. Polynomials are represented by capital letters such as

$$X(z) = \sum_{n=0}^{N-1} x_n z^n. \tag{1.1}$$

For transforms, we use the notation $\bar{X}_k$, which, for a DFT, has the form

$$\bar{X}_k = \sum_{n=0}^{N-1} x_n W^{nk}. \tag{1.2}$$

We have also sometimes adopted Rader's notation $\langle x \rangle_p$ for the residue of x modulo p.

1.3 The Structure of the Book

Chapter 2 presents introductory material on number theory and polynomial algebra. This covers in an intuitive way various topics such as the divisibility of integers and polynomials, congruences, roots defined in finite fields and rings. This background in mathematics is required to understand the rest of the book and may be skipped by the readers who are already familiar with number theory and modern algebra.

Fast convolution algorithms are discussed in Chap. 3. It is shown that most of these algorithms can be represented in polynomial algebra and can be considered as various forms of nesting.

The fourth chapter gives a simple development of the conventional fast Fourier transform algorithm and presents new versions of this method such as the Rader-Brenner algorithm.

Chapter 5 is devoted to the computation of discrete Fourier transforms by convolutions and deals primarily with Winograd Fourier transform which is an extremely efficient algorithm for the computation of the discrete Fourier transform.

In Chaps. 6 and 7, we introduce the polynomial transforms which are DFTs defined in finite rings and fields of polynomials. We show that these transforms are computed without multiplications and provide an efficient tool for computing multidimensional convolutions and DFTs.

In Chap. 8, we turn our attention to algorithms implemented in modular arithmetic and we present the number theoretic transforms which are DFTs defined in finite rings and fields of numbers. We show that these transforms may have important applications when implemented in special purpose hardware.

2. Elements of Number Theory and Polynomial Algebra

Many new digital signal processing algorithms are derived from elementary number theory or polynomial algebra, and some knowledge of these topics is necessary to understand these algorithms and to use them in practical applications.

This chapter introduces the necessary background required to understand these algorithms in a simple, intuitive way, with the intent of familiarizing engineers with the mathematical principles that are most frequently used in this book. We have made here no attempt to give a complete rigorous mathematical treatment but rather to provide, as concisely as possible, some mathematical tools with the hope that this will prompt some readers to study further, with some of the many excellent books that have been published on the subject [2.1–4].

The material covered in this chapter is divided into two main parts: elementary number theory and polynomial algebra. In elementary number theory, the most important topics for digital signal processing applications are the Chinese remainder theorem and primitive roots. The Chinese remainder theorem, which yields an unusual number representation, is used for number theoretic transforms (NTT) and for index manipulations which serve to map one-dimensional problems into multidimensional problems. The primitive roots play a key role in the definition of NTTs and are also used to convert discrete Fourier transforms (DFT) into convolutions, which is an important step in the development of the Winograd Fourier transform algorithm.

In the polynomial algebra section, we introduce briefly the concepts of rings and fields that are pervasive throughout this book. We show how polynomial algebra relates to familiar signal processing operations such as convolution and correlation. We introduce the Chinese remainder theorem for polynomials and we present some complexity theory results which apply to convolutions and correlations.

2.1 Elementary Number Theory

In this section, we shall be essentially concerned with the properties of integers. We begin with the simple concept of integer division.

2.1.1 Divisibility of Integers

Let a and b be two integers, with b positive. The division of a by b is defined by

$$a = bq + r, \qquad 0 \leqslant r < b, \tag{2.1}$$

where q is called the *quotient* and r is called the *remainder*. When $r = 0$, b and q are *factors* or divisors of a, and b is said to *divide a*, this operation being denoted by $b \mid a$. When a has no other divisors than 1 and a, a is a *prime*. In all other cases, a is *composite*.

When a is composite, it can always be factorized into a product of powers of prime numbers $p_i^{c_i}$, where c_i is a positive integer, with

$$a = \prod_i p_i^{c_i}. \tag{2.2}$$

The *fundamental theorem of arithmetic* states that this factorization is unique.

The largest positive integer d which divides two integers a and b is called the *greatest common divisor* (GCD) and denoted

$$d = (a, b); \tag{2.3}$$

when $d = (a, b) = 1$, a and b have no common factors other than 1 and they are said to be *mutually prime* or *relatively prime*.

The GCD can be found easily by a division algorithm known as the *Euclidean algorithm*. In discussing this algorithm, we shall assume that a and b are positive integers. This is done without loss of generality, since $(a, b) = (-a, b) = (a, -b) = (-a, -b)$. Dividing a by b yields

$$a = bq_1 + r_1, \quad r_1 < b \tag{2.4}$$

by definition, $d = (a, b) \leqslant a$ or b. Therefore, if $r_1 = 0$, $b \mid a$ and $(a, b) = b$. If $r_1 \neq 0$, we obtain, by continuation of this procedure, the following system of equations:

$$\begin{aligned}
b &= r_1 q_2 + r_2, \qquad r_2 < r_1 \\
r_1 &= r_2 q_3 + r_3, \qquad r_3 < r_2 \\
&\cdots\cdots\cdots\cdots \\
r_{k-2} &= r_{k-1} q_k + r_k, \quad r_k < r_{k-1} \\
r_{k-1} &= r_k\, q_{k+1}
\end{aligned} \tag{2.5}$$

Since $r_1 > r_2 > r_3 \ldots$, the last remainder is zero. Thus, by the last equation, $r_k \mid r_{k-1}$. The preceding equation implies that $r_k \mid r_{k-2}$, since $r_k \mid r_{k-1}$. Finally, we obtain $r_k \mid b$ and $r_k \mid a$. Hence, r_k is a divisor of a and b. Suppose now that c is any divisor of a and b. By (2.4), c also divides r_1 [illegible] implies that c divides r_2, $r_3 \ldots r_k$. Thus, any divisor c of a and b divides r_k and therefore $c \leqslant r_k$. Hence, r_k is the GCD of a and b.

An important consequence of Euclid's algorithm is that the GCD of two integers a and b is a linear combination of a and b. This can be seen by rewriting (2.4) and (2.5) as

$$\begin{aligned} r_1 &= a - bq_1 \\ r_2 &= b - r_1 q_2 \\ &\cdots\cdots\cdots \\ r_k &= r_{k-2} - r_{k-1}\, q_k. \end{aligned} \tag{2.6}$$

The first equation shows that r_1 is a linear combination of a and b. The second equation shows that r_2 is a linear combination of b and r_1 and therefore of both a and b. Finally, the last equation implies that r_k is a linear combination of a and b. Since $r_k = (a, b)$, we have

$$(a, b) = ma + nb, \tag{2.7}$$

where m and n are integers. When a and b are mutually prime, (2.7) reduces to Bezout's relation

$$1 = ma + nb. \tag{2.8}$$

We now change our point of view by considering a linear equation with integer coefficients a, b, and c

$$ax + by = c \tag{2.9}$$

where x and y are a pair of integers which are the solution of this *Diophantine equation*. Such an equation has a solution if and only if $(a, b) | c$. To demonstrate this point, we note the following. It is obvious from (2.9) that for $a = 0$ or $b = 0$, we must have $b | c$ or $a | c$.

For $a \neq 0$, $b \neq 0$, it is apparent that if (2.9) holds for integers x and y, then $d = (a, b)$ is such that $d | c$. Conversely, if $d | c$, $c = c_1 d$ and (2.7) implies the existence of two integers m and n such that $d = ma + nb$. Hence $c = c_1 d = c_1 ma + c_1 nb$, and the solutions of the Diophantine equation are given by $x = c_1 m$, $y = c_1 n$. Thus, for $(a, b) | c$, the solution of the Diophantine equation is given by the Euclidean algorithm. The solution of the Diophantine equation is not unique, however. This can be seen by considering a particular solution $c = ax_0 + by_0$. Assuming x, y is another solution, we have

$$a(x - x_0) = b(y_0 - y) \tag{2.10}$$

and, by dividing this expression by d, we obtain

$$(a/d)(x - x_0) = (b/d)(y_0 - y) \tag{2.11}$$

Since $[(a/d), (b/d)] = 1$, this implies that $(b/d) | (x - x_0)$ and $x = x_0 + (b/d)s$ where s is an integer. Substituting into (2.11), we obtain

$$\begin{aligned} y &= y_0 - (a/d)\, s \\ x &= x_0 + (b/d)\, s. \end{aligned} \tag{2.12}$$

This defines a class of linearly related solutions for (2.9) which depend upon the integer s.

As a numerical example, consider the equation

$$15x + 9y = 21.$$

We first use Euclid's algorithm to determine the GCD d with $a = 15$ and $b = 9$,

$$\begin{aligned} 15 &= 9 \cdot 1 + 6 \\ 9 &= 6 \cdot 1 + 3 \\ 6 &= 3 \cdot 2 \end{aligned}$$

Hence $d = 3$. Since $3 | 21$, the Diophantine equation has a solution. We now define 3 as a linear combination of 15 and 9 by recasting the preceding set of equations as

$$\begin{aligned} 6 &= 15 - 9 \cdot 1 \\ 3 &= 9 - 6 \cdot 1 = -15 + 2 \cdot 9 \end{aligned}$$

Thus, $m = -1$ and $n = 2$. Dividing $c = 21$ by $d = 3$ yields $c_1 = 7$. This gives a particular solution $x_0 = -7$, $y_0 = 14$. If we divide $a = 15$ and $b = 9$ by $d = 3$, we obtain $(a/d) = 5$ and $(b/d) = 3$. Hence, the general solution to the Diophantine equation becomes

$$\begin{aligned} y &= 14 - 5s \\ x &= -7 + 3s, \end{aligned}$$

where s is any integer.

2.1.2 Congruences and Residues

In (2.1), the division of an integer a by an integer b produces a remainder r. All integers a which give the same remainder when divided by b can be thought as pertaining to the same *equivalence class* relative to the equivalence relation $a = bq + r$.

Two integers a_1 and a_2 pertaining to the same class are said to be *congruent modulo* b and the equivalence is denoted

$$a_1 \equiv a_2 \quad \text{modulo } b. \tag{2.13}$$

Thus, two numbers a_1 and a_2 are congruent modulo b if

$$b | (a_1 - a_2). \tag{2.14}$$

Underlying the concept of congruence is the fact that, in many physical problems, one is primarily interested in measuring relative values within a given range. This is apparent, for instance, when measuring angles. In this case, the angles are defined from 0 to 359° and two angles that differ by a multiple of 360° are considered to be equal. Hence angles are defined modulo 360.

Thus, in congruences, we are interested only in the remainder r of the division of a by b. This remainder is usually called the *residue* and is denoted by

$$r \equiv a \quad \text{modulo } b. \tag{2.15}$$

This representation is sometimes simplified to a form with the symbol $\langle\ \rangle$ [2.4],

$$r = \langle a \rangle_b, \tag{2.16}$$

where the subscript is omitted when there is no ambiguity on the nature of the modulus b.

It follows directly from the definition of residues given by (2.14) that additions and multiplications can be performed directly on residues

$$\begin{aligned} \langle a_1 + a_2 \rangle &= \langle\langle a_1 \rangle + \langle a_2 \rangle\rangle \\ \langle a_1 a_2 \rangle &= \langle\langle a_1 \rangle \langle a_2 \rangle\rangle. \end{aligned} \tag{2.17}$$

With congruences, division is not defined. We can however define something close to it by considering the linear congruence

$$a\,x \equiv c \quad \text{modulo } b \tag{2.18}$$

This linear congruence is the Diophantine equation $ax + by = c$ in which all terms are defined modulo b. Thus, we know by the results of the preceding section that we can find values of x satisfying (2.18) if and only if $d \mid c$, with $d = (a, b)$. In this case, the solutions can be derived from (2.12) and are given by

$$x \equiv x_0 + (b/d)s \quad \text{modulo } b, \tag{2.19}$$

where x_0 is a particular solution and s can be any integer smaller than b. However, there are only d distinct solutions since $(b/d)s$ has only d distinct values modulo b. An important consequence of this point is that the linear congruence $ax \equiv c$ modulo b always has a unique solution when $(a, b) = 1$. Thus, when $(a, b) \mid c$, the linear congruence $ax \equiv c$ modulo b can be solved and Euclid's algorithm provides a method for computing the values of x which satisfy this relation. We shall see later that Euler's theorem gives a more elegant solution to the linear congruence (2.18) when $(a, b) = 1$.

We consider now the problem of solving a set of simultaneous linear congruences with different moduli. Changing our notation, we want to find the integer x which satisfies simultaneously the k linear congruences