Abstract Algorithms for Multidimensional Discrete Fourier Transforms

ausgeführt am Institut für Angewandte und Numerische Mathematik der Technischen Universität Wien

unter der Anleitung von Prof. Dr. Christoph Überhuber

durch

Florian Kaltenberger
Matrikelnummer: 9725857
Pramergasse 8/12
1090 Wien.

Wien, 21. Mai 2002
Abstract

The discrete Fourier transform (DFT) of large multidimensional datasets is an essential tool for many fields in science and engineering. The standard approach to compute the multidimensional DFT (MDFT) is to apply one-dimensional FFTs along each dimension (row-column method). Although optimized one-dimensional FFT routines can be used, for large data sets this method is very ineffective, since it requires transpositions of the whole data array. Especially in parallel environments, where communication is very slow compared to computation, this method is almost impracticable.

In this work alternative algorithms for computing the MDFT are explored. These algorithms are based on the abstract definition of the discrete Fourier transform, which removes the dependence on coordinates and solely references the additive Abelian group structure of the indexing set. This definition unifies all additive one- and multidimensional FFT algorithms, including the Cooley-Tukey and Good-Thomas FFT.

Special emphasis is put on reduced transform algorithms (RTAs), which—in their most general form—provide a full utilization of the duality between periodic and decimated data in the Fourier transform. The cases where the indexing set \( A \) is two-dimensional and of the form \( \mathbb{Z}^* \times \mathbb{Z}^* \), or \( \mathbb{Z}^* \times \mathbb{Z}^* \), or \( d \)-dimensional and of equal and prime size in at least \( d - k + 1 \) dimensions, are discussed in more detail.

RTAs tend to minimize interprocessor communication and are therefore very well suited for parallel implementations. Hybrid algorithms, that nest RTAs into other multidimensional FFT algorithms, can be used to get a finer trade-off between degree of parallelism and granularity as compared to stand-alone RTAs.
Zusammenfassung


In der vorliegenden Arbeit werden alternative Algorithmen zur Berechnung der MDFT vorgestellt und untersucht. Diese Algorithmen beruhen auf der abstrakten Definition der Fourier-Transformation, die koordinatenunabhängig ist und nur die additive Abelsche Gruppenstruktur der Indexmenge verwendet. Diese Definition vereint alle additiven eindimensionalen und mehrdimensionalen FFT Algorithmen, einschließlich der Cooley-Tukey- und der Good-Thomas-FFT.

Das Hauptthema dieser Arbeit sind “reduced transform algorithms” (RTAs), die in ihrer allgemeinsten Form eine Anwendung der Dualität zwischen periodischen und dezimierten Daten in der Fourier-Transformation sind. Jene Fälle, wo die Indexmenge A zweidimensional und von der Form \( /p \times \) /p, oder \( /p^r \times /p^s \), oder d-dimensional und zumindest in \( d-k+1 \) Dimensionen gleich groß und prim ist, werden genauer besprochen.

RTAs reduzieren im Allgemeinen die Interprozessorkommunikation und eignen sich deshalb hervorragend für eine parallele Implementierung. Hybride Algorithmen, die RTAs in andere multidimensionale FFT Algorithmen einbetten, können verwendet werden um den Grad der Parallelität des Algorithmus auf den des Computers abzustimmen.
Preface

The discrete Fourier transform (DFT) is one of the principal algorithmic tools in the field of scientific computing as DFT methods can be used to solve various scientific and engineering problems. For example, the DFT is essential to the digital processing of analogous signals. DFT methods are also used to solve partial differential equations which arise, for example, in the field of computational fluid dynamics. Especially in the fields of seismology, X-ray crystallography, radar, sonar and medical imaging, DFTs of large multidimensional data sets are required.

The DFT requires the evaluation of a special matrix-vector product. Thus, the order of complexity of the DFT was long thought to be $O(N^2)$. In 1965 Cooley and Tukey [5] published an algorithm, the fast Fourier transform (FFT), which reduced the computational complexity of the DFT to $O(N \log N)$. Since then numerous studies have been published on how the FFT can be implemented efficiently on advanced computer systems. The first step was made by Pease [18] in 1968. He introduced an algorithm well suited for implementations on parallel computers. In his pioneering paper, which was not based on the Cooley-Tukey variant, Pease used the Kronecker product notation to describe the FFT. This formalism is particularly useful as mathematical formulas involving Kronecker product operations are easily translated into various programming constructs.

By algebraically manipulating Kronecker product formulas, different programs that achieve the same computation but have different performance characteristics can be obtained. In this way algorithms can be made architecture adaptive to better match specific computer architectures.

In a multidimensional setting, there is an added degree of freedom in viewing the space for which to compute the Fourier transform. This freedom is exploited best, when regarding the multidimensional Fourier transform (MDFT) in a more abstract setting, one that removes the dependence on coordinates and solely references the additive Abelian group structure of the indexing set. This approach highlights the fundamental role played by the duality between periodization and decimation in MDFT algorithm design.

The main emphasis in this work is put on reduced transform algorithms (RTAs), a class of algorithms that reduce the problem of calculating the multidimensional Fourier transform to a set of lower-order Fourier transforms. Since these lower-order Fourier transforms are completely independent parallel computation is feasible. Several types of reduced transform algorithms are explored and their implementation is discussed.
Synopsis

Chapter 1 gives an introduction into the history of fast Fourier transform (FFT) algorithms along with some prototypical applications. Furthermore, two approaches on how to derive FFT algorithms are outlined.

Chapter 2 reviews tensor products and the algebra of stride permutations. The latter plays a fundamental role in establishing rules in form of tensor product identities. The Cooley-Tukey FFT algorithm for one-dimensional DFTs and the vector-radix FFT algorithm for multidimensional DFTs are explained.

Chapter 3 summarizes the mathematics required for the abstract definition of the Fourier transform. Finite Abelian groups and the concept of duality play a central role in this theory.

Chapter 4 defines the Fourier transform of a finite Abelian group. This definition removes the dependence on coordinates as it solely references the additive Abelian group structure of the indexing set. This approach highlights the fundamental role played by the duality between periodization and decimation in multidimensional FFT algorithm design.

Chapter 5 states theorems that unify all additive one-dimensional and multidimensional FFT algorithms, including the Cooley-Tukey and the Good-Thomas FFT. These results can be used to compute FFTs of different dimensions using the same algorithm. Such an algorithm is called a dimensionless FFT.

Chapter 6 depicts a special collection of subgroups of finite Abelian groups and their relationship to duality. These subgroups are the building blocks of reduced transform algorithms.

Chapter 7 explains the reduced transform algorithm (RTA) and provides detailed examples for some special cases.

Chapter 8, finally, discusses the parallel implementation of RTAs and the possibility to nest RTAs into other FFT algorithms. This approach is especially useful in parallel environments.
Acknowledgments

First of all I would like to express my gratitude to my advisor Christoph Ueberlueber and to Franz Franchetti for making this work possible, for their guidance, their encouragement, their support, and their valuable suggestions. They cleared up many misconceptions that retarded my working progress and they facilitated my participation in current research projects.

I’m most grateful to my friend Nina Rygalyk for her encouragement in bad times, for sharing the good times with me and for her love throughout the years. Many thanks to Georg Langs and Gregor Meller for proofreading this document and for their critical comments. Thanks also to my mathematics high school teacher Gottfried Obereder, who has opened up the world of mathematics to me and always believed in me.

I have become acquainted with several colleagues during my studies with whom it was a pleasure to cooperate with. Some of them have become very good friends and I would like to express my thanks to all of them for their support. I especially enjoyed to explore interesting mathematical details and Vienna’s nightlife together with them at our weekly “Stammtisch” meetings.

I would like to thank the Linux community for providing me with a decent operating system and typesetting system. Not to forget the library of the Vienna Technical University, which was always a place of inspiration and a harbor for my thoughts.

Last but not least, I would like to thank my parents for making my studies possible, for their assistance, advice and support. Without their aid I would have never gotten this far.
Contents

1 Introduction ......................................................... 1
  1.1 History and Evolution of the FFT ......................... 1
  1.2 Derivation of FFT Algorithms .......................... 3
    1.2.1 Standard Approach .......................... 3
    1.2.2 Abstract Approach ........................ 3
  1.3 Applications of FFT algorithms ....................... 5

2 Kronecker Products and the FFT ................................. 7
  2.1 Kronecker Products .................................. 7
    2.1.1 Algebraic properties of Kronecker products .... 8
    2.1.2 Kronecker Products and Parallel Programming .... 9
  2.2 Stride Permutations .................................. 11
    2.2.1 Algebraic Properties of Stride Permutations .... 12
  2.3 One-Dimensional FFT Algorithms ..................... 13
  2.4 Multidimensional FFT Algorithms .................. 16
    2.4.1 The Row-Column Algorithm .................. 17
    2.4.2 The Vector-Radix FFT ....................... 18

3 Finite Abelian Groups ............................................ 21
  3.1 Basic Definitions and Properties .................... 21
  3.2 The Character Group ................................ 23
  3.3 Duality .............................................. 27
  3.4 The Chinese Remainder Theorem .................... 29
  3.5 The Vector Space $L(A)$ .......................... 33

4 Fourier Transforms of Finite Abelian Groups .................. 34
  4.1 The Fourier Transform of $A$ ....................... 34
  4.2 The Induced Fourier Transform .................... 36
  4.3 Periodic and Decimated Data ....................... 37
  4.4 Periodization and Decimation ....................... 40
5 Abstract Multidimensional FFT Algorithms  . . . . . . . . . .  43
   5.1 The Abstract Good-Thomas FFT  . . . . . . . . . . . . . .  43
   5.2 The Abstract Cooley-Tukey FFT  . . . . . . . . . . . . . .  45
   5.3 The Dimensionless FFT  . . . . . . . . . . . . . . . . . . .  47

6 Lines and Planes on Abelian Groups and their Duals  . . .  51
   6.1 Lines  . . . . . . . . . . . . . . . . . . . . . . . . . . . . .  51
       6.1.1 The Prime Case  . . . . . . . . . . . . . . . . . . . .  52
       6.1.2 The Power of Prime Case  . . . . . . . . . . . . . .  54
   6.2 Duals of Lines  . . . . . . . . . . . . . . . . . . . . . . .  58
   6.3 Hyperplanes  . . . . . . . . . . . . . . . . . . . . . . . .  59

7 Reduced Transform Algorithms  . . . . . . . . . . . . . . . .  63
   7.1 General Structure of RTAs  . . . . . . . . . . . . . . . . .  63
       7.1.1 The Standard RTA  . . . . . . . . . . . . . . . . . . .  64
       7.1.2 The Dual RTA  . . . . . . . . . . . . . . . . . . . . .  64
   7.2 Examples  . . . . . . . . . . . . . . . . . . . . . . . . . .  65
       7.2.1 2-D RTA for $A = . /p \times . /p$, $p$ Prime  . . . .  65
       7.2.2 2-D RTA for $A = . /p^r \times . /p^r$, $p$ Prime  . .  69
       7.2.3 $d$-dimensional RTA  . . . . . . . . . . . . . . . . . .  72

8 Implementation of FFTs on Parallel Architectures  . . . . . .  74
   8.1 Dataflow in Parallel FFTs  . . . . . . . . . . . . . . . . .  74
   8.2 Machine Model  . . . . . . . . . . . . . . . . . . . . . . .  75
   8.3 Parallel RC Algorithms  . . . . . . . . . . . . . . . . . .  75
   8.4 Parallel RTAs  . . . . . . . . . . . . . . . . . . . . . . .  76
       8.4.1 An RTA for the 2-D Prime Case  . . . . . . . . . . . .  77
   8.5 Hybrid Algorithms  . . . . . . . . . . . . . . . . . . . . .  79
       8.5.1 The Cooley-Tukey/RTA Hybrid Algorithm  . . . . . .  79
       8.5.2 The Good-Thomas/RTA Hybrid Algorithm  . . . . . .  82

9 Conclusion  . . . . . . . . . . . . . . . . . . . . . . . . . . .  86
Chapter 1

Introduction

1.1 History and Evolution of the FFT

The first use of trigonometric series in analysis can be found in the work of L. Euler (1707–1783). Euler presented the formulas for the coefficients of the Fourier series representation of a function of a real variable. Euler used trigonometric series to describe sound propagation in an elastic medium.

The stature of Euler in his own time assured that his work was read by his contemporaries, particularly a number of French mathematicians. Among them was A.-C. Clairaut (1713–1765), who published the earliest formula for the discrete Fourier transform in 1754. His formula was restricted, however, to a cosine-only finite Fourier series. J. L. Lagrange (1736–1813) published a formula for sine-only series in 1762. D. Bernoulli (1700–1782) expressed the form of a vibrating string as a series of sine and cosine terms with arguments of both time and distance in 1753. This implied that an arbitrary function could be expressed as an infinite sum of cosines.

Clairaut and Lagrange were concerned with orbital mechanics and the problem of determining the details of an orbit from a finite set of observations. C. F. Gauss (1777–1855) extended the work of Euler and Lagrange dealing with trigonometric interpolation to include periodic functions which are not necessarily odd or even. This was done while considering the problem of determining the orbit of certain asteroids from sample locations. Gauss developed an algorithm similar to the Cooley-Tukey fast Fourier transform, which was based on the reduction of one large Fourier transform to several smaller ones for computing the coefficients of a finite Fourier series. Gauss' treatise describing the algorithm appeared in his collected works as an unpublished manuscript. The presumed year of the creation of his algorithm is 1805. Though not influencing the work of Cooley and Tukey, the roots of the FFT algorithm date back to the early nineteenth century!

J. B. J. Fourier's (1768–1830) interest in heat conduction led him to begin work in 1807 on the "analytic theory of heat". Published in 1822, it shows how a mathematical series of sine and cosine terms can be used to analyze heat conduction in solid bodies. Fourier spent the rest of his life working on it and expanding it to include the Fourier integral.

Both the Fourier series and the Fourier integral allow to transform physically realizable time-domain waveforms to the frequency domain and vice versa. Today
many areas of science benefit from Fourier analysis. However the method did not gain acceptance in the time of J.B. J. Fourier. The reason behind this was the distrust in the use of infinite series. One influential mathematician in 1828 commented: “Divergent series are an invention of the devil, and it is shameful to base on them any demonstration whatsoever”. Since then work by Dirichlet, Friedrich, Riemann, and others have resolved any doubts about the validity of the Fourier series.

The major setback in using the Fourier transform was its prohibitive computational effort. The discrete Fourier transform (DFT) required extensive computational time to evaluate. Techniques to reduce the computational effort were developed, for instance, by C. Runge in 1903, who essentially described the FFT. Danielson and Lanczos in 1942 recognized certain symmetries and periodicities which reduced the number of operations.

Even with the advent of the digital computer the techniques to reduce the computational complexity of the DFT were unknown until 1965 when J.W. Cooley and J.W. Tukey published their DFT algorithm which became known as the fast Fourier transform (FFT).

The success story of this discovery began at a meeting of the US president’s scientific advisory committee. R.L. Garwin, who was in desperate need of a fast means to compute Fourier transforms for his research with helium, noted that Tukey was writing algorithms for Fourier transforms and asked him to outline his techniques. Garwin then went to the IBM research center to have it programmed. Cooley, a relatively new member of the staff, was given the problem because to his own admission “had nothing important to do” and quickly worked it out. Thinking he would hear no more about it, Cooley “went back to doing some real work”. This was obviously not the case since Garwin foresaw a wide range of applications and widely publicized the results.

The FFT idea has been re-discovered several times in the past two centuries, before becoming a well-known standard. This is partly due to differing languages and notations.

**Today’s Challenges**

“Multidimensional signal processing is a child of the computer revolution” was stated in Tolimieri et al. [24]. Several fields of applications require multidimensional data for complete and faithful modeling. The massive-size data sets typically found in multidimensional signal processing place intensive demands on the handling of the communication aspects of the computation. Memory utilization—transfer of data between main storage and computation units—must be precisely matched to the communication capabilities of the target processor.

Multidimensional computations offer a wider range of possible implementations
compared to one-dimensional computations due to a greater degree of freedom in the data indexing set. This freedom is exploited best, when regarding the multidimensional Fourier transform (MDFT) in a more abstract setting, one that removes the dependence on coordinates, and solely references the additive Abelian group structure of the indexing set. This approach highlights the fundamental role played by the duality between periodization and decimation in MDFT algorithm design.

1.2 Derivation of FFT Algorithms

1.2.1 Standard Approach

The standard approach to describe DFT algorithms is as a matrix-vector product, which in turn can be described (i) by sums notation, or (ii) by utilizing the Kronecker product.

For a long time the sum notation was thought to be the only mathematical representation of DFT algorithms. Thus, one had to put up with indices containing complicated terms that were rather confusing. Therefore, this methodology on the whole makes it difficult to understand the general structure without careful and intensive studies.

In 1968, finally, Pease [18] realized the existence of an alternative notation. He was the first who utilized Kronecker products for describing FFT algorithms. So it was possible to express all required operations on the matrix level (in contrast to the matrix-entry level before) and to obtain considerably clearer structures.

During the following twenty-five years, however, not many authors made use of this technique. Ultimately, the Kronecker product formalism is more and more accepted since it makes it easy to modify FFT algorithms by exploiting the underlying algebraic structure.

Van Loan used the Kronecker product technique for a state of the art presentation of FFT algorithms in his remarkable book “Computational Frameworks for the Fast Fourier Transform” [25].

In the twenty-five years between the publications of Pease and Van Loan, only a few authors used this powerful technique: Temperton [21] and Johnson et al. [12] for FFT implementations on classic vector computers and Norton and Silberger [17] on parallel computers with MIMD architecture.

1.2.2 Abstract Approach

The abstract approach to developing DFT algorithms makes use of the definition of the discrete Fourier transform on finite Abelian groups. This is especially
1. Introduction

useful for multidimensional Fourier transforms, since it removes the dependence on the coordinates. Any discrete periodic function can be viewed as a function from a finite Abelian group into the set of complex numbers. For an Abelian group \( A \) the set of these complex-valued functions, which is a vector space, will be denoted by \( L(A) \). Then the DFT is just a linear mapping \( F \) from \( L(A) \) into \( L(A^*) \), where \( A^* \) denotes the character group of \( A \), which is isomorphic to \( A \).

The Cooley-Tukey algorithm (Cooley and Tukey [5]) and its variants can be generalized to apply to an arbitrary finite Abelian group. This algorithm computes the Fourier transform of a finite Abelian group in terms of the Fourier transform of an arbitrary subgroup and on its quotient group.

Recently, Auslander et al. [2, 3] introduced a way of constructing Cooley-Tukey algorithms that uses an arbitrary choice of coset representatives for the quotient group. Different choices of coset representatives lead to different data flow in the algorithm and different twiddle factors. These results generalize all of the known Cooley-Tukey type algorithms and provide new algorithms with alternative data flow.

If the order of a finite Abelian group \( N = o(A) \) is a product of two relatively prime integers \( N = PQ \), then the abstract definition of the Fourier transform leads to the \textit{Good-Thomas prime factor algorithm} (PFA) (Good [9]). This algorithm is similar in structure to the Cooley-Tukey algorithm, but no longer requires twiddle factor multiplications.

\textit{Reduced transform algorithms} (RTAs) (An et al. [1], Tolimieri et al. [22, 24]) are methods for computing multidimensional Fourier transforms by decomposing the Fourier transform on the finite Abelian group \( A \) into Fourier transforms on lines (or more generally on hyperplanes) passing through the origin that cover \( A \). Here a line in a finite Abelian group \( A \) is a maximal cyclic subgroup of \( A \). By using the decimation-periodization theorem, which will be developed in Chapter 4, it can be demonstrated that an RTA computes the Fourier transform on \( A \). A detailed discussion of RTAs is given in Chapter 7.

Another class of DFT algorithms also makes use of the multiplicative structure of the indexing set. These methods are therefore called \textit{multiplicative Fourier transform algorithms}. For example, Rader’s algorithm (Rader [19]) for \( p \)-point Fourier transforms, where \( p \) is a prime, uses the fact that \( \mathbb{Z}/p \) is a field and the unit group \( U(p) \) is cyclic. Reordering input and output data corresponding to a generator of \( U(p) \), the \( p \)-point DFT can be computed by a \((p - 1)\)-point cyclic convolution. Multiplicative Fourier transform algorithms, however, are not subject of this work. For more information consult Tolimieri et al. [23, 24].
1.3 Applications of FFT algorithms

FFT algorithms are ubiquitous tools in science and engineering. The following examples try to evoke some feeling for the universal applicability of FFTs.

**Seismic Observation.** One specific application of the discrete Fourier transform was to test, whether a seismic activity was caused by an earthquake or by a nuclear bomb. These events can be distinguished in the frequency domain, because the respective spectra have strongly differing characteristics.

There are more seismic applications of the FFT. The oil industry, for example, uses the FFT as a fundamental exploration tool. With the so-called “seismic reflection method” it is possible to map sub-surface sedimentary rock layers.

**Filtering.** Filters are used to attenuate or enhance certain frequency components of a given signal. There are low pass, high pass and band pass filters. But there are even more filter issues that can be addressed with FFTs, for example the complexity reduction of finite impulse responses by providing filter convolution.

**Image Processing.** Filters can also be applied to two- and more-dimensional signals like digital pictures. They can be smoothed, sharp edges can be enhanced as well as disturbing background noise can be reduced. Especially for high-noise medical images (like some X-ray pictures) the FFT can be used to enhance quality and visibility. Furthermore, the spectral representation of a digital picture can be used to gain valuable information for pattern-recognition purposes.

Advanced image processing techniques are used also for fingerprint analysis. Using FFTs, the frequencies corresponding to the regular lines of a check can be found and deleted, minimizing distortion. With the background removed, the lines that make up the fingerprint become visible.\(^1\)

**Data Communication.** The FFT is usually associated with low level aspects of communication. For instance, to understand how a signal will behave when sent through communication channels, amplifiers etc. Especially the degree of distortion can be modeled easily knowing the bandwidth of the channel and the spectrum of the signal. For example, if a signal is made up of certain frequency components but only part of them pass through the channel, adding up only those components passing results in a good approximation to the distorted signal.

**Astronomy.** Sometimes it is not possible to get all the required information from a “normal” telescope. In such situations radio waves or radar are used instead of light. The radar signals are treated just like any other time varying voltage signal and can be processed digitally. For example, the satellite Magellan, released in 1989 and sent to earth's closest planet, Venus, was equipped with modern radar

\(^1\)See, for instance, [http://www.mediacy.com/notes/an130.htm](http://www.mediacy.com/notes/an130.htm).
and digital signal processing capabilities and provided excellent data. These data were used to create a computer-generated virtual flight above planet Venus.\(^2\)

**Optics.** In optical theory the signal is the oscillatory electric field at a point in space where light passes by. The Fourier transform of the signal is equivalent to breaking up the light into its components by using a prism. The Fourier transform is also used to calculate the diffracted intensity with the experiments of light passing through narrow slits (even Young’s famous double slit experiment made use of the Fourier Transform). These ideas can be applied to all kinds of wave analysis applications like acoustic, X-ray, and microwave diffraction.

**Speech Recognition.** The field of digital speech processing and recognition is a multi-million Euro business by now. Advances in computational speed and new language models have made speech recognition possible on simple PCs. Today, speech recognition is a very wide field, consisting of isolated word recognition for highly specialized fields as medicine as well as connected word and even conversational speech recognition.

The FFT plays a small, but important role in the recognition process. It is used to transform the signal, usually after a first filtering process, into the frequency domain to obtain its spectrum, because the critical features for perceiving speech by the human ear are mainly included in the spectral information, while the phase information does not play a crucial role.\(^3\)

**X-ray Crystallography.** In crystallographic structure analysis, the FFT is used to derive the structure of a crystal from its X-ray diffraction pattern. The electron density function \(\rho(r)\) in a crystal determines its diffraction pattern and conversely. The function \(\rho(r)\) is a triply periodic function of the position vector \(r\), and consequently, can be expanded in a Fourier series. The coefficients of the Fourier series are the structure factors and their magnitudes are determined from the X-ray diffraction pattern. Since the structure factors are in general complex, the determination of the structure of a crystal is equivalent to finding the phases of the structure factors: the phase problem of X-ray crystallography. Hauptman [10], discusses the phase problem and shows that, due to the atomicity of crystal structures and the redundancy of observed magnitudes, the problem is, in principle, solvable. In practice, this means that for small structures the phase problem is directly solvable. However, for larger structures indirect methods are used. Indirect methods involve computing the Fourier transform and its inverse many times to calculate the structure factors from their magnitudes.

\(^3\)More on speech recognition at http://coral.lili.uni-bielefeld.de/Classes/Summer96/Acoustic/acoustic2/acoustic.html.
Chapter 2

Kronecker Products and the FFT

This chapter introduces the formalisms of Kronecker products (tensor products) and stride permutations, which are the foundations of most standard FFT algorithms. Kronecker products are a powerful tool for keeping track of index calculations and for establishing simple rules, in the form of Kronecker product identities, that can be used to modify an algorithm for optimal performance as data size and target architecture vary. The proofs will mostly be omitted and can be found in van Loan [25].

Sections 2.3 and 2.4 give the standard definitions of one-dimensional and multidimensional DFTs. FFT algorithms using Kronecker product notation are derived. In the one-dimensional case this will be the Cooley-Tukey FFT algorithm and in the multidimensional case this will be the vector-radix algorithm.

2.1 Kronecker Products

Definition 2.1.1 (Kronecker Product)  The Kronecker product (tensor product) of the matrices $A \in M_{1 \times N_1}$ and $B \in M_{2 \times N_2}$ is the block structured matrix

$$A \otimes B := \begin{pmatrix} a_{0,0}B & \cdots & a_{0,N_1-1}B \\ \vdots & \ddots & \vdots \\ a_{M_1-1,0}B & \cdots & a_{M_1-1,N_1-1}B \end{pmatrix} \in M_{M_1 M_2 \times N_1 N_2}.$$

Denote by $\mathcal{e}_m^M$, $m = 0, 1, \ldots, M - 1$ the vector in $M$ with a 1 in the $m$th component and 0 elsewhere. The set of vectors

$$\{ \mathcal{e}_m^M : m = 0, 1, \ldots, M - 1 \}$$

is the standard basis of $M$. Set $M = M_1 M_2$ and form the set of tensor products

$$\mathcal{e}_{m_1}^{M_1} \otimes \mathcal{e}_{m_2}^{M_2}, \quad m_1 = 0, 1, \ldots, M_1 - 1, \quad m_2 = 0, 1, \ldots, M_2 - 1.$$  

(2.2)

Since

$$\mathcal{e}_{m_2 + m_1 M_2}^{M} = \mathcal{e}_{m_1}^{M_1} \otimes \mathcal{e}_{m_2}^{M_2}, \quad m_1 = 0, 1, \ldots, M_1 - 1, \quad m_2 = 0, 1, \ldots, M_2 - 1,$$

(2.3)
the set (2.2) ordered by choosing \( m_2 \) to be the fastest running parameter is the standard basis of \( M \). In particular, the set of tensor products of the form
\[
x^{M_1} \otimes y^{M_2}
\]
spans \( M \).
Most of the tensor product identities can be proved by computing the action of both sides on the standard basis (2.2).

2.1.1 Algebraic properties of Kronecker products

Property 2.1.1 (Associativity) If \( A, B, C \) are arbitrary matrices, then
\[
(A \otimes B) \otimes C = A \otimes (B \otimes C).
\]
Thus, the expression \( A \otimes B \otimes C \) is unambiguous.

Property 2.1.2 (Transposition) If \( A, B \) are arbitrary matrices, then
\[
(A \otimes B)^T = A^T \otimes B^T.
\]

Property 2.1.3 (Inversion) If \( A, B \) are regular matrices, then
\[
(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}.
\]

Property 2.1.4 (Mixed-Product Property) If \( A, B, C, D \) are arbitrary matrices, then
\[
(A \otimes B)(C \otimes D) = AC \otimes BD,
\]
provided the products \( AC \) and \( BD \) are defined.

A consequence of this property is the following factorization.

Corollary 2.1.1 (Decomposition) If \( A \in M_{M_1 \times N_1} \) and \( B \in M_{M_2 \times N_2} \), then
\[
A \otimes B = AI_{N_1} \otimes I_{N_2} B = (A \otimes I_{M_2})(I_{M_1} \otimes B),
\]
\[
A \otimes B = I_{M_1} A \otimes BI_{N_2} = (I_{M_1} \otimes B)(A \otimes I_{N_2}).
\]
The mixed-product property can be generalized in two different ways (for matrices of appropriate size):
\[
(A_1 \otimes A_2 \otimes \cdots \otimes A_k)(B_1 \otimes B_2 \otimes \cdots \otimes B_k) = A_1B_1 \otimes A_2B_2 \cdots \otimes A_kB_k,
\]
and
\[
(A_1 \otimes B_1)(A_2 \otimes B_2) \cdots (A_k \otimes B_k) = (A_1A_2 \cdots A_k) \otimes (B_1B_2 \cdots B_k).
\]
Property 2.1.5 (Distributive Law) If $A$, $B$, $C$, are arbitrary matrices, then

$$(A + B) \otimes C = (A \otimes C) + (B \otimes C),$$

$$A \otimes (B + C) = (A \otimes B) + (A \otimes C).$$

The Kronecker product is not commutative. This non-commutativity is mainly responsible for the richness of the tensor product algebra, and naturally leads to a distinguished class of permutations, the stride permutations. The first important consequence of this lack of commutativity can be seen in the relationship between tensor product and matrix direct sum

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix},$$

where the $0$'s represent blocks of zeros of size necessary to form the matrix.

Property 2.1.6 (Left Distributive Law) It holds that

$$(A \oplus B) \otimes C = (A \otimes C) \oplus (B \otimes C).$$

The right distributive law does not hold.

### 2.1.2 Kronecker Products and Parallel Programming

A connection between Kronecker products and computer architectures can be established by associating special types of Kronecker products with particular types of processor and/or memory organization.

Let $B \in \mathbb{L}^{L \times L}$ and $I_M \in \mathbb{M}^{M \times M}$ be the identity matrix. An expression of the form

$$I_M \otimes B = \begin{pmatrix} B & 0 \\ B & 0 \\ \vdots & \vdots \\ B & 0 \end{pmatrix},$$

can be viewed as a parallel operation. Its action on a vector $x$ can be performed by computing the independent action of $B$ on each of the $M$ consecutive segments of $x$ of size $L$. Parallel operations can be implemented efficiently on parallel architectures.

**Example 2.1.1** Let $B_2 \in \mathbb{L}^{2 \times 2}$ be an arbitrary matrix and let $I_3 \in \mathbb{M}^{3 \times 3}$ be the identity matrix. Then

$$y := (I_3 \otimes B_2)x$$
is given by

\[
\begin{pmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5
\end{pmatrix}
\begin{pmatrix}
b_{0,0} & b_{0,1} \\
b_{1,0} & b_{1,1}
\end{pmatrix}
\begin{pmatrix}
b_{0,0} & b_{0,1} \\
b_{1,0} & b_{1,1}
\end{pmatrix}
\begin{pmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{pmatrix}
\]

This matrix-vector product can be realized by splitting the input vector \(x \in \mathbb{R}^6\) into 3 subvectors of length 2 and performing the respective matrix-vector products

\[
\begin{pmatrix}
y_0 \\
y_1
\end{pmatrix}
:=
\begin{pmatrix}
b_{0,0} & b_{0,1} \\
b_{1,0} & b_{1,1}
\end{pmatrix}
\begin{pmatrix}
x_0 \\
x_1
\end{pmatrix}
\]

\[
\begin{pmatrix}
y_2 \\
y_3
\end{pmatrix}
:=
\begin{pmatrix}
b_{0,0} & b_{0,1} \\
b_{1,0} & b_{1,1}
\end{pmatrix}
\begin{pmatrix}
x_2 \\
x_3
\end{pmatrix}
\]

\[
\begin{pmatrix}
y_4 \\
y_5
\end{pmatrix}
:=
\begin{pmatrix}
b_{0,0} & b_{0,1} \\
b_{1,0} & b_{1,1}
\end{pmatrix}
\begin{pmatrix}
x_4 \\
x_5
\end{pmatrix}
\]

Let \(A \in \mathbb{R}^{M \times M}\) and \(I_L \in \mathbb{R}^{L \times L}\) be the identity matrix. An expression of the form

\[
A \otimes I_L =
\begin{pmatrix}
a_{0,0}I_L & \cdots & a_{0,M-1}I_L \\
\vdots & \ddots & \vdots \\
a_{M-1,0}I_L & \cdots & a_{M-1,M-1}I_L
\end{pmatrix}
\]

can be viewed as a vector operation. To compute its action on a vector \(x \in \mathbb{R}^N\), \(x\) is segmented into \(M\) consecutive segments of size \(L\),

\[
x_0, x_1, \ldots, x_{M-1},
\]

and the vector operations

\[
a_{m,0}x_0 + a_{m,1}x_1 + \cdots + a_{m,M-1}x_{M-1}, \quad m = 0, 1, \ldots, M - 1
\]

are performed. Since these operations may be performed by a vector processor, expressions of the form \(A \otimes I_L\) are called vector operations.

**Example 2.1.2** Let \(A_2 \in \mathbb{R}^{2 \times 2}\) and let \(I_3 \in \mathbb{R}^{3 \times 3}\) be the identity matrix. Then

\[
y := (A_2 \otimes I_3)x
\]

is given by

\[
\begin{pmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5
\end{pmatrix}
:=
\begin{pmatrix}
a_{0,0} & a_{0,1} \\
a_{0,0} & a_{0,1} \\
a_{1,0} & a_{1,1} \\
a_{1,0} & a_{1,1} \\
\end{pmatrix}
\begin{pmatrix}
x_0 \\
x_1 \\
x_2 \\
x_3 \\
x_4 \\
x_5
\end{pmatrix}
\]
This matrix-vector product can be computed by splitting the vector $x \in \mathbb{C}^N$ into 2 subvectors of length 3 and performing single scalar multiplications with these subvectors:

$$
\begin{pmatrix}
y_0 \\
y_1 \\
y_2 \\
y_3 \\
y_4 \\
y_5 \\
\end{pmatrix}
:=
\begin{pmatrix}
a_{0,0} & x_0 & x_1 & x_2 & x_3 & x_4 & x_5 \\
0 & a_{1,0} & x_1 & x_2 & x_3 & x_4 & x_5 \\
0 & 0 & a_{1,1} & x_1 & x_2 & x_3 & x_5 \\
\end{pmatrix}
$$

2.2 Stride Permutations

Consider a vector $x \in \mathbb{C}^N$, where $N = LM$. Segment the vector into $M$ vectors of size $L$

$$x_0, x_1, \ldots, x_{M-1},$$

where $x_m$ is formed from the $L$ components of $x$ beginning with $x(mL)$. Form the $L \times M$ matrix

$$\text{Mat}_{L \times M}(x) = [x_0 x_1 \cdots x_{M-1}].$$

The $l,m$ coefficient of $\text{Mat}_{L \times M}(x)$ is given by

$$x(l,m) = x(l + mL), \quad l = 0, 1, \ldots, L-1, \quad m = 0, 1, \ldots, M-1.$$

Conversely, if $X$ is an $L \times M$ matrix, the lexicographic ordering on $X$ is the ordering on the elements of $X$ defined by running in order down the columns of $X$. Denote by $x \in \mathbb{C}^N$ the vector determined by the lexicographic ordering on $X$. Clearly, the vector corresponding to $\text{Mat}_{L \times M}(x)$ is $x$ itself. Denote the transpose of $\text{Mat}_{L \times M}(x)$ by $\text{Mat}^T_{L \times M}(x)$.

**Definition 2.2.1 (Stride Permutation)** For a vector $x \in \mathbb{C}^N$, $N = ML$ the stride permutation $P(N, L)$ is defined by

$$y = P(N, L)x,$$

where $y$ is the vector corresponding to the matrix

$$\text{Mat}^T_{L \times M}(x).$$

The permutation operator $P(N, L)$ sorts the components of $x$ according to their index modulo $L$. Thus, components with indices equal to $0 \mod L$ come first, followed by the components with indices equal to $1 \mod L$, and so on.

**Definition 2.2.2 (Even-Odd Sort Permutation)** The permutation $P(N, 2)$ ($N$ even) is called an even-odd sort permutation, because it groups the even-indexed and odd-indexed components together.
Definition 2.2.3 (Perfect Shuffle Permutation) The permutation $P(N, \frac{N}{2})$ ($N$ even) is called a perfect shuffle permutation, since its action on a deck of cards could be the shuffling of two equal piles of cards so that the cards are interleaved one from each pile.

Because of its importance, the perfect shuffle permutation $P(N, N/2)$ is denoted in short by $\Pi(N)$.

The action of $P(N, L)$ on a vector of the form $a^M \otimes b^L$, where $a^M \in \mathbb{R}^M$ and $b^L \in \mathbb{R}^L$ can also be described as

$$P(N, L)(a^M \otimes b^L) = b^L \otimes a^M.$$

2.2.1 Algebraic Properties of Stride Permutations

Property 2.2.1 (Identity)

$$P(1, M) = P(M, M) = I_M$$

Property 2.2.2 (Inversion/Transposition) If $N = ML$,

$$P(N, L)^{-1} = P(N, L)^T = P(N, M)$$

Property 2.2.3 (Multiplication) If $N = MLK$,


Example 2.2.1

$$P(2^i, 2)^{-1} = P(2^i, 2^{i-1}) = \Pi(2^i).$$

As already mentioned, the Kronecker Product is not commutative. However, with the aid of stride permutations, the order of factors can be reverted.

Theorem 2.2.1 (Commutation) If $A \in \mathbb{R}^{M \times R}$ and $B \in \mathbb{R}^{L \times S}$, then setting $N = LM$ and $T = RS$,

$$P(N, L)(A \otimes B) = (B \otimes A)P(T, S).$$

Proof: Johnson et al. [12].

Several special cases are worth noting. Suppose $M = R$ and $S = L$. Then $N = T$, and

$$P(N, L)(A \otimes B)P(N, L)^{-1} = B \otimes A.$$

Application of this leads to

$$I_L \otimes A = P(N, L)(A \otimes I_L)P(N, M),$$

$$B \otimes I_M = P(N, L)(I_M \otimes B)P(N, M).$$


Stride permutations interchange parallel and vector operations. The readdressing prescribed by \( P(N, M) \) on input and \( P(N, L) \) on output turns the vector operation \( A \otimes I_L \) into the parallel operation \( I_L \otimes A \) and the parallel operation \( I_M \otimes B \) into the vector operation \( B \otimes I_M \). Continuing this way, it is possible to write

\[
A \otimes B = (A \otimes I_L)(I_M \otimes B) = P(N, M)(I_L \otimes A)P(N, L)(I_M \otimes B), \tag{2.4}
\]

which can be used to compute the action of \( A \otimes B \) as a sequence of two parallel operations. It also holds that

\[
A \otimes B = (A \otimes I_L)P(N, M)(B \otimes I_M)P(N, L), \tag{2.5}
\]

which can be used to compute the action of \( A \otimes B \) as a sequence of two vector operations. The stride permutations intervene between computational stages, providing a mathematical language for describing the readdressing.

Occasionally it will be necessary to permute the factors in a tensor product of more than two factors. The following permutation generalizes the stride permutation.

**Definition 2.2.4 (Digit Permutation)** Let \( N = N_1 N_2 \cdots N_R \) and let \( \sigma \) be a permutation of the numbers 1, 2, \ldots, \( R \). Then the digit permutation is defined by

\[
P_{\sigma}(N_1, \ldots, N_R)(e^{N_1}_{i_1} \otimes \cdots \otimes e^{N_R}_{i_R}) = (e^{N_{\sigma(1)}}_{i_{\sigma(1)}} \otimes \cdots \otimes e^{N_{\sigma(R)}}_{i_{\sigma(R)}}).
\]

**Theorem 2.2.2 (Permutation)** Let \( A_r, r = 0, 1, \ldots, R \), be an \( N_r \times N_r \) matrix, and let \( \sigma \) be a permutation of the numbers 1, 2, \ldots, \( R \). Then,

\[
A_1 \otimes \cdots \otimes A_R = P_{\sigma}(N_1, \ldots, N_R)^{-1}(A_{\sigma(1)} \otimes \cdots \otimes A_{\sigma(R)})P_{\sigma}(N_1, \ldots, N_R).
\]

**Proof:** Johnson et al. [12].

\[
\square
\]

### 2.3 One-Dimensional FFT Algorithms

In this section, the tensor product formulation of one-dimensional Cooley-Tukey FFT algorithms will be reviewed.

**Definition 2.3.1 (Fourier Transform Matrix)** The \( N \)-point Fourier transform matrix \( F(N) \) is defined by

\[
F(N) = [\omega^{jk}]_{j,k=0,1,\ldots,N-1}, \quad \omega = e^{2\pi i/N}.
\]
Definition 2.3.2 (Discrete Fourier Transform) The DFT vector

\[ y = (y_0, \ldots, y_{N-1})^\top \in \mathbb{C}^N \]

of the data vector

\[ x = (x_0, \ldots, x_{N-1})^\top \in \mathbb{C}^N \]

is defined by the special matrix-vector product \( y = F(N)x \).

Several properties of the \( N \)-point Fourier transform matrix are useful in derivations and applications. First, \( F(N) \) is a Vandermonde matrix defined by the first row (the top row consists of all 1’s and will be referred to as the 0th row),

\[ 1, \omega, \omega^2, \ldots, \omega^{N-1}. \]

As it will be discussed in more detail in Chapter 4, the rows of \( F(N) \) are the additive characters of the Abelian group \( \mathbb{Z}/N \). The sum of the components in every row but the 0th row is equal to 0. The sum of the components in the 0th row is \( N \). With these properties in mind it is easy to show that

\[ F(N)F(N)^* = NI_N, \]

where * denotes complex conjugation. Since \( F(N)^T = F(N) \), it follows that \( F(N)^{-1} = \frac{1}{N}F(N)^* \).

The additive FFT can be derived using the following properties of \( F(N) \). Setting

\[ \Omega_{N/2} := \text{diag}(1, \omega, \ldots, \omega_{N/2-1}^{N/2-1}), \quad N \text{ even}, \]

the four symmetry conditions \((k, j = 0, 1, \ldots, N/2 - 1)\)

\[
\begin{align*}
[F(N)\Pi(N)]_{k,j} &= \omega_N^{k(2j)} = \omega_{N/2}^{kj} = [F(N/2)]_{k,j} \\
[F(N)\Pi(N)]_{k+N/2,j} &= \omega_N^{(k+N/2)(2j)} = \omega_{N/2}^{(k+N/2)j} = [F(N/2)]_{k,j} \\
[F(N)\Pi(N)]_{k,j+N/2} &= \omega_N^{k(2j+1)} = \omega_N^{k} \omega_{N/2}^{kj} = [\Omega_{N/2}F(N/2)]_{k,j} \\
[F(N)\Pi(N)]_{k+N/2,j+N/2} &= \omega_N^{(k+N/2)(2j+1)} = -\omega_N^{k(2j+1)} = [-\Omega_{N/2}F(N/2)]_{k,j}
\end{align*}
\]

imply the \textit{radix-2 splitting}:

\[ F(N) \Pi(N) = \begin{pmatrix} F(N/2) & \Omega_{N/2}F(N/2) \\ F(N/2) & -\Omega_{N/2}F(N/2) \end{pmatrix}, \]

These relations can be easily established by simple computations due to the fact that \( \omega_N^2 = \omega_{N/2} \) and \( \omega_N^{N/2} = -1 \).
The term “radix-2 splitting” indicates that this relation establishes a connection between the full-sized DFT matrix \( F(N) \) and the half-sized DFT matrix \( F(N/2) \). Recursive application of this splitting process is the heart of all radix-2 FFT algorithms.

Thus, \( F(N) \) can be factorized according to

\[
F(N) = \begin{pmatrix}
I_{N/2} & I_{N/2} \\
I_{N/2} & -I_{N/2}
\end{pmatrix}
\begin{pmatrix}
I_{N/2} & 0 \\
0 & \Omega_{N/2}
\end{pmatrix}
\begin{pmatrix}
F(N/2) \\
F(N/2)
\end{pmatrix} P(N, 2).
\]

This factorization can be expressed in terms of Kronecker products.

**Theorem 2.3.1 (Cooley-Tukey Factorization)** For \( N \) even, \( N \geq 2 \)

\[
F(N) = (F(2) \otimes I_{N/2}) T(N) (I_2 \otimes F(N/2)) P(N, 2),
\]

\[
T(N) = (I_{N/2} \oplus \Omega_{N/2}),
\]

For \( N = 2^n \), repeated application of Theorem 2.3.1 (Cooley-Tukey factorization) leads to the following factorization (Johnson et al. [12]).

**Theorem 2.3.2 (Cooley-Tukey Radix-2 FFT)** It holds that

\[
F(2^n) = \prod_{i=1}^{n} (I_{2^{i-1}} \otimes F(2) \otimes I_{2^{n-i}})(I_{2^{i-1}} \otimes T(2^{n-i+1})) \right] R_{2^n},
\]

where

\[
R_{2^n} := \prod_{i=2}^{n} (I_{2^{n-i}} \otimes P(2^i, 2)).
\]

The permutation matrix \( R_{2^n} \) is called a bit reversal matrix.

By using the Kronecker product property 2.1.4 (multiplicative property) the expression

\[
(I_{2^{i-1}} \otimes F(2) \otimes I_{2^{n-i}})(I_{2^{i-1}} \otimes T(2^{n-i+1}))
\]

can be written as

\[
I_{2^{i-1}} \otimes ((F(2) \otimes I_{2^{n-i}})(I_{2^{n-i}} \oplus \Omega_{2^{n-i}})),
\]

where

\[
(F(2) \otimes I_{2^{n-i}})(I_{2^{n-i}} \oplus \Omega_{2^{n-i}}) = \begin{pmatrix}
I_{2^{n-i}} & \Omega_{2^{n-i}} \\
I_{2^{n-i}} & -\Omega_{2^{n-i}}
\end{pmatrix} =: B_{2^{n-i+1}}.
\]

The matrix \( B_{2^{n-i+1}} \) is called a radix-2 Butterfly matrix.

Thus, (2.7) can be expressed as
\[ F(2^n) = \left[ \prod_{i=1}^{n} (I_{2^{i-1}} \otimes B_{2^{n-i+1}}) \right] R_{2^n}. \] (2.8)

More generally, if \( p \) divides \( N \), it is possible to relate \( F(N) \) to \( F(N/p) \) by analogous considerations.

**Theorem 2.3.3 (Fundamental Radix-\( p \) Splitting)** For \( N = pq \geq 2 \)
\[ F(N) = (F(p) \otimes I_q)T_q(N)(I_p \otimes F(q))P(N,p), \] (2.9)
with
\[ T_q(N) := \text{diag}(I_q, \Omega_{p,q}, \ldots, \Omega_{p,q}^{q-1}), \] (2.10)
where
\[ \Omega_{p,q} := \text{diag}(1, \omega_{N}, \ldots, \omega_{N}^{q-1}). \]

The Cooley-Tukey factorization (2.6) is also called *decimation in time* (DIT) splitting, since the time sampled data is first divided into parts to which, secondly, FFTs of appropriate lengths are applied. A *decimation in frequency* (DIF) splitting can be obtained by transposing the Cooley-Tukey factorization. Using the algebraic properties of Kronecker products and stride permutations, even more variations of the Cooley-Tukey factorization (2.6) can be derived (see, for example Johnson et al. [12]).

### 2.4 Multidimensional FFT Algorithms

**Definition 2.4.1 (\( n \)-dimensional Fourier Transform)** For positive integers \( N_1, N_2, \ldots, N_n \) and an \( n \)-dimensional array
\[ A(j_1, j_2, \ldots, j_n), \quad j_i \in \mathbb{Z}/N_i, \quad i = 1, 2, \ldots, n, \]
the \( N_1 \times N_2 \times \cdots \times N_n \) \( n \)-dimensional discrete Fourier transform of \( A \), denoted by \( F(N_1, N_2, \ldots, N_n) \) \( A \), is the \( n \)-dimensional array \( B \) defined by
\[ B(k_1, \ldots, k_n) = \sum_{j_1=0}^{N_1-1} \cdots \sum_{j_n=0}^{N_n-1} A(j_1, \ldots, j_n) \omega_1^{k_1 j_1} \cdots \omega_n^{k_n j_n}, \] (2.11)
where \( k_i \in \mathbb{Z}/N_i, \quad i = 1, 2, \ldots, n, \) and \( \omega_i \) is the \( N_i \)th root of unity.

Alternatively, (2.11) can be interpreted as the matrix-vector product
\[ b = (F_{N_1} \otimes F_{N_2} \otimes \cdots \otimes F_{N_n})a, \] (2.12)
where \( a \) and \( b \) are vectors of size \( N = N_1 N_2 \cdots N_n \) obtained by ordering the elements of \( A \) and \( B \) lexicographically by dimension.

In the following subsections the *row-column* and the *vector-radix* algorithm for computing the multidimensional Fourier transform (2.11) will be reviewed.
2.4.1 The Row-Column Algorithm

The computation of (2.11) can be decomposed into a sequence of \( n \) one-dimensional Fourier transforms:

\[
A_1(j_1, \ldots, j_{n-1}, k_n) = \sum_{j_n=0}^{N_n-1} A(j) \omega_n^{j_n j_n},
\]

\[
A_2(j_1, \ldots, k_{n-1}, k_n) = \sum_{j_n=0}^{N_n-1} A_1(j_1, \ldots, j_{n-1}, k_n) \omega_n^{j_n-1 j_n-1},
\]

\[
\vdots
\]

\[
B(k) = \sum_{j_n=0}^{N_n-1} A_{n-1}(j_1, k_2, \ldots, k_n) \omega_1^{k_1 j_1}.
\]  

Equation (2.13) is also referred to as the \textit{row-column method}, since in the two-dimensional case it is equivalent to computing \( B \) by a sequence of one-dimensional FFTs of the columns of \( A \) followed by a sequence of one-dimensional FFTs of the resulting rows.

The row-column method requires data transposition between each stage of the computation. This can be seen by using (2.4) to write the two-dimensional \( N_1 \times N_2 \) FFT as

\[
F(N_1) \otimes F(N_2) = P(N_1 N_2, N_1)(I_{N_2} \otimes F(N_1))P(N_1 N_2, N_2)(I_{N_1} \otimes F(N_2)).
\]  

First \( N_1 \) \( N_2 \)-point FFTs are computed on the rows of the two-dimensional array, then the rows and columns are interchanged, and \( N_2 \) \( N_1 \)-point FFTs are again performed on the rows of the array. Finally, the processed data is output by a second row-column interchange. For large-size problems, carrying out these transpositions by transferring between main and local memory can be the most time-consuming part of the computation.

Another way to compute the multidimensional Fourier transform (2.12) is to substitute any one-dimensional FFT algorithm, including any one-dimensional multiplicative DFT algorithm in the form of a matrix factorization, in place of \( F(N_r) \). Tensor product identities, especially the multiplicative theorem of tensor products, can be used to generate a wide range of algorithms. For example, substituting the Cooley-Tukey factorization (2.6) into (2.12) leads to the \textit{vector-radix method}, which will be derived in the following subsection.
2.4.2 The Vector-Radix FFT

For simplicity of notation, the following discussion will be restricted to two dimensions. The general case can be derived analogously.

The two-dimensional vector-radix FFT is obtained by taking the tensor product of two one-dimensional FFT algorithms. Let \( M = M_1 M_2 \) and \( L = L_1 L_2 \) and write

\[
F(M) = (F(M_1) \otimes I_{M_2}) T_{M_1}(M) (I_{M_1} \otimes F(M_2)) P(M, M_1),
\]

\[
F(L) = (F(L_1) \otimes I_{L_2}) T_{L_1}(L) (I_{L_1} \otimes F(L_2)) P(L, L_1).
\]

Then

\[
F(M) \otimes F(L) = X_2 TX_1 P,
\]

where

\[
X_1 = I_{M_1} \otimes F(M_2) \otimes I_{L_1} \otimes F(L_2),
\]

\[
X_2 = F(M_1) \otimes I_{M_2} \otimes F(L_1) \otimes I_{L_2},
\]

\[
P = P(M, M_1) \otimes P(L, L_1),
\]

\[
T = T_{M_1}(M) \otimes T_{L_1}(L).
\]

The actions of the matrices \( P, X_1, T \) and \( X_2 \) of the factorization (2.15) will now be discussed in more detail.

\( P \) is a two-dimensional stride permutation, since it describes the readdressing that permits interchange between parallel and vector two-dimensional operations. The action of \( P \) on \( A \) is given by the matrix product

\[
P(L, L_2) A P(M, M_1).
\]

The matrix product \( P(L, L_2) A \) reorders the rows of \( A \) by stride \( L_2 \), while the product \( AP(M, M_1) \) reorders the columns of \( A \) by stride \( M_2 \).

The case

\[
P(L, 2) A P(M, M/2), \quad 2 | l, \ 2 | M,
\]

is called the two-dimensional inverse perfect shuffle, which can be written as

\[
\begin{pmatrix}
A_{0,0} & A_{0,1} \\
A_{1,0} & A_{1,1}
\end{pmatrix},
\]

where \( A_{0,0} \) consists of all even/even indices, \( A_{0,1} \) of all even/odd indices, \( A_{1,0} \) of all odd/even indices and \( A_{1,1} \) of all odd/odd indices.

The matrix

\[
P(L, L/2) A P(M, 2), \quad 2 | l, \ 2 | M
\]

is called the two-dimensional perfect shuffle.
Example 2.4.1 The action of $P(M, 2) \otimes P(L, L/2)$ on an $8 \times 8$ array results in the $8 \times 8$ array

\[
\begin{pmatrix}
0,0 & 0,4 & 0,1 & 0,5 & 0,2 & 0,6 & 0,3 & 0,7 \\
4,0 & 4,4 & 4,1 & 4,5 & 4,2 & 4,6 & 4,3 & 4,7 \\
1,0 & 1,4 & 1,1 & 1,5 & 1,2 & 1,6 & 1,3 & 1,7 \\
5,0 & 5,4 & 5,1 & 5,5 & 5,2 & 5,6 & 5,3 & 5,7 \\
2,0 & 2,4 & 2,1 & 2,5 & 2,2 & 2,6 & 2,3 & 2,7 \\
6,0 & 6,4 & 6,1 & 6,5 & 6,2 & 6,6 & 6,3 & 6,7 \\
3,0 & 3,4 & 3,1 & 3,5 & 3,2 & 3,6 & 3,3 & 3,7 \\
7,0 & 7,4 & 7,1 & 7,5 & 7,2 & 7,6 & 7,3 & 7,7 \\
\end{pmatrix}
\]

The matrix $X_1$ can be rewritten as

\[
X_1 = B(I_{M_1L_1} \otimes F(M_2) \otimes F(L_2))B^{-1},
\]
where $B$ is a permutation called a block permutation. The action of $B$ on the a matrix $A$ can be described by the following two steps.

1. Partition the $L \times M$ matrix $A$ into $M_1L_1$ contiguous submatrices of size $L_2 \times M_2$.

2. Order the elements of $A$ by placing the lexicographic ordering on the collection of submatrices and on the elements of the submatrices.

Regard the array $A$ as a vector $a$ by placing the lexicographic ordering on $A$ and $B$ as a one-dimensional permutation on $a$. Then the factor $I_{M_1L_1} \otimes F(M_2) \otimes F(L_2)$ acts in parallel on the $M_1L_1$ contiguous segments of $Ba$. Since these one-dimensional segments correspond to the $M_1L_1$ contiguous subblocks of $A$, this is equivalent to computing the two-dimensional operation $F(M_2) \otimes F(L_2)$ on each of these subblocks of $A$. The inverse permutation $B^{-1}$ reorders the resulting $L \times M$ array lexicographically.

Thus, $X_1$ is a two-dimensional operation and acts on $A$ by computing in parallel the $L_1 \times M_1$ two-dimensional Fourier transforms $F(L_1, M_1)$ on the $L_2M_2$ contiguous subblocks of size $L_1 \times M_1$ of $A$, placing the results back in place. Matrices of this form are called parallel two-dimensional operations.

The matrix $T$ is a diagonal matrix and acts on $A$ by multiplying the elements of $A$ with the appropriate twiddle factors.

The matrix $X_2$ can be rewritten as

\[
X_2 = B(F(M_1) \otimes F(L_1) \otimes I_{M_2L_2})B^{-1}
\]
where $B$ is the same matrix as above. Thus, $X_2$ acts on $A$ by computing the $L_2 \times M_2$ two-dimensional Fourier transform $F(L_2, M_2)$ on the $L_2M_2$ contiguous subblocks of size $L_1 \times M_1$ of $A$, placing the results back in place. Matrices of this form are called vector two-dimensional operations.
Using equations (2.16) and (2.17), equation (2.15) can be rewritten as

\[ B(F(M_1) \otimes F(L_1) \otimes I_{M_2L_2})B^{-1}TB(I_{M_1L_1} \otimes F(M_2) \otimes F(L_2))B^{-1}P. \]  

(2.18)

Since \( B^{-1}TB \) is a diagonal matrix, this factorization has the same form as the Cooley-Tukey theorem (Theorem 2.3.1), and the vector-radix algorithm is a two-dimensional analog of the Cooley-Tukey theorem (Johnson and Johnson [11]).

In fact, there are many generalizations to the Cooley-Tukey theorem that apply to multidimensional FFTs, all of which can be stated in a single theorem about the Fourier transform of finite Abelian groups, which will be the topic of the next two chapters. The theorem itself will be stated and proved in Chapter 5.
Chapter 3

Finite Abelian Groups

This chapter summarizes some results from algebra that are necessary for defining the discrete Fourier transform on a finite Abelian group. This is in turn the foundation for designing many multidimensional FFT algorithms. The presentation follows mainly Tolimieri et al. [24].

3.1 Basic Definitions and Properties

First, some fundamental definitions and facts are given.

Definition 3.1.1 (Group) A set $A$ together with a binary operation $+$ is called a group, if (i) $+$ is associative, (ii) there exists a neutral element $e \in A$ and (iii) for all $x \in A$ there exits an inverse $x^{-1} \in A$.

Definition 3.1.2 (Abelian Group) A group $A$ is called Abelian or commutative if $+$ is commutative.

Definition 3.1.3 (Finite Group) A group $A$ is called finite if $A$ is finite.

Definition 3.1.4 (Order of a Group) For a group $A$, let $o(A) = |A|$ denote the group order and for $a \in A$ let $o(a) = |\{na : n \in \mathbb{Z} \}|$ denote the order of $a$.

Definition 3.1.5 (Generated Subgroup) Let $A$ be a group and $a \in A$. Then

$$\text{gp}(a) = \{na : n \in \mathbb{Z} \},$$

is the subgroup generated by $a$.

Definition 3.1.6 (Cyclic Group) A group $A$ is called cyclic, if and only if there exists an $a \in A$ such that $A = \text{gp}(a)$.

Cyclic groups can be characterized by the order of their generators.

Lemma 3.1.1 Let $A = \text{gp}(a)$ be a cyclic group generated by $a$. Then there are two cases:

(i) If $o(a) = \infty$, then $A$ is infinite, and $A = \{e, a, -a, 2a, -2a, \ldots \}$. 

21
(ii) If \( o(a) = N \in \mathbb{Z} \), then \(|A| = N\), and \( A = \{e, a, 2a, \ldots, (N - 1)a\}\).

**Proof:** Lang [16]. □

This means that every cyclic group is either isomorphic to \( \mathbb{Z}/N\mathbb{Z} \) or isomorphic to \( \mathbb{Z}/N\mathbb{Z} \), depending on the generator of the group.

**Corollary 3.1.2** If \( o(a) = \infty \) then \( \text{gp}(a) \) is isomorphic to \( \mathbb{Z} \), and if \( o(a) = N \in \mathbb{Z} \), then \( \text{gp}(a) \) is isomorphic to \( \mathbb{Z}/N\mathbb{Z} \), where \( \mathbb{Z}/N\mathbb{Z} \) are integers modulo \( N \).

**Proof:** Lang [16]. □

**Definition 3.1.7 (Group Direct Product)** For groups \( B_1, B_2, \ldots, B_R \), the group direct product

\[
B_1 \times B_2 \times \cdots \times B_R
\]

is the set of \( R \)-tuples

\[
b = (b_1, b_2, \ldots, b_R), \quad b_r \in B_r, \quad r = 1, 2, \ldots, R.
\]

under componentwise addition.

If \( A \) is a finite Abelian group and there are subgroups \( A_1, A_2, \ldots, A_R \), such that \( A = A_1 + A_2 + \cdots + A_R \) and \( A_{i+1} \cap (A_1 + \cdots + A_i) = \{0\}, \quad i = 1, 2, \ldots, R - 1 \) then instead of writing \( A = A_1 \times A_2 \times \cdots \times A_R \), the notation

\[
A = A_1 \oplus A_2 \oplus \cdots \oplus A_R,
\]

is used. \( A \) is said to be the direct sum of \( A_1, A_2, \ldots, A_R \).

**Theorem 3.1.3 (Fundamental Theorem of Finite Abelian Groups)** Any finite Abelian group is isomorphic to a direct product of cyclic groups of prime power order. Any two such decompositions have the same number of factors of each order.

**Proof:** Lang [16]. □

Since every cyclic group of order \( N \) is isomorphic to \( \mathbb{Z}/N\mathbb{Z} \), Theorem 3.1.3 is equivalent to the following proposition.

**Corollary 3.1.4** If \( A \) is a finite Abelian group of order \( N \) then there exist integers \( N_1, N_2, \ldots, N_R \) such that \( A \) is isomorphic to the group direct product

\[
\mathbb{Z}/N_1 \times \mathbb{Z}/N_2 \times \cdots \times \mathbb{Z}/N_R,
\]

where \( N = N_1 N_2 \cdots N_R \) is a factorization of \( N \).
Proof: Lang [16]

Every finite Abelian group can serve as the indexing set for a multidimensional Fourier transform. In the above, \( A \) is the indexing set for the \( R \)-dimensional Fourier transform of size \( N_1 \times \cdots \times N_R \).

**Definition 3.1.8 (Presentation)** A group direct product of the form (3.2), which is isomorphic to \( A \), is called a presentation of \( A \).

The choice of a presentation is not unique. A finite Abelian group can usually be written in several ways as the direct product of cyclic groups. The number of cyclic group factors as well as their respective sizes can vary. The importance of non-uniqueness, brought out in the following chapters, can briefly be described as follows.

“A Fourier transform will be assigned to each finite Abelian group. Each presentation produces a standard multidimensional Fourier transform whose dimension and size are fixed by the presentation. It will be shown [...] that multidimensional Fourier transforms corresponding to different presentations differ solely by automorphisms of the finite Abelian group. An algorithm computing the multidimensional Fourier transform relative to one presentation can be used to compute the multidimensional Fourier transform relative to any other presentation. Several FFT algorithms, including the Good-Thomas algorithm, will be a straightforward consequence of this remark.” (Tolimieri et al. [24])

### 3.2 The Character Group

A finite Abelian group \( A \) can be viewed as the time parameter space on which signals are defined and \( A^* \) may be considered to be the frequency parameter space on which Fourier transforms are defined. However, it will turn out that \( A^* \) is isomorphic to \( A \).

Denote the multiplicative group of complex numbers with magnitude 1 by \( U \) and the subgroup of all complex \( N \)th roots of unity by \( U_N \). \( U_N \) is a cyclic group of order \( N \) having the element \( e^{2\pi i/N} \) as a generator. In fact any element of the form

\[
e^{2\pi in/N}, \quad \gcd(n, N) = 1,
\]

generates \( U_N \). If \( M \) divides \( N \), then \( U_M \) is a subgroup of \( U_N \). If \( N = N_1N_2 \) and \( N_1 \) and \( N_2 \) are relatively prime \( U_N \) can be written as a direct product

\[
U_N = U_{N_1} \times U_{N_2}.
\]
Definition 3.2.1 (Character of a Group) A character $a^*$ of $A$ is a group homomorphism of $A$ into $U$:

$$a^*(a + b) = a^*(a)a^*(b), \quad a, b \in A.$$ 

$a^*(a)$ will be denoted by $\langle a, a^* \rangle$.

Lemma 3.2.1 Every character of $A$ is a homomorphism of $A$ onto $U_N$.

Proof: Since $A$ has order $N$, $Na = 0$ for all $a \in A$ and

$$\langle a, a^* \rangle^N = \langle Na, a^* \rangle = 1, \quad a \in A,$$

for every character $a^*$ of $A$. Thus

$$\langle a, a^* \rangle \in U_N, \quad a \in A,$$

and every character of $A$ is a homomorphism of $A$ into $U_N$. \qed

Definition 3.2.2 (Character Group) Denote the set of all characters of $A$ by $A^*$. $A^*$ is called the character group of $A$, where an Abelian group addition is defined on $A^*$ by

$$\langle a, a^* + b^* \rangle = \langle a, a^* \rangle \langle a, b^* \rangle, \quad a^*, b^* \in A^*, a \in A.$$ 

Example 3.2.1 ($A = . /N$) The group $A$ is cyclic of order $N$. Characters of $A$ are completely determined by their action on any generator, say the element $1$ of $A$. Set $u = e^{2\pi i/N}$. $U_N$ is generated by $u$. There are $N$ distinct characters of $A$, each determined by the power of $u$ onto which the generator of $A$ is mapped. The characters can be defined using the group $A$ as an indexing set. $A^* = \{\chi_a : a \in A\} = \{\chi_i : i = 0, 1, \ldots N - 1\}$ where $\chi_i$ is defined by $\langle 1, \chi_i \rangle = u^i$.

For $a \in A$ define the mapping $\psi(a)$ of $A$ onto $U$ by the rule

$$\langle b, \psi(a) \rangle = u^{ab}, \quad b \in A.$$ 

The power $u^{ab}$ is well-defined, since $a$ and $b$ are defined modulo $N$. Direct computation shows that $\psi(a) \in A^*$ and that the mapping

$$\psi : A \to A^*$$

is an isomorphism of $A$ onto $A^*$.

Example 3.2.2 ($A = . /N_1 \times . /N_2, \quad N = N_1N_2$) The elements

$$e_1 = (1, 0), \quad e_2 = (0, 1)$$

generate $A$. A typical element $a \in A$ can be written uniquely as

$$a = (a_1, a_2) = a_1e_1 + a_2e_2, \quad a_1 \in . /N_1, \quad a_2 \in . /N_2.$$
Characters of $A$ are determined uniquely by their actions on $e_1$ and $e_2$. Set

$$u_1 = e^{2\pi i/N_1}, \quad u_2 = e^{2\pi i/N_2}.$$ 

$U_{N_1}$ and $U_{N_2}$ are generated by $u_1$ and $u_2$.

Characters must map $e_1$ into $U_{N_1}$ and $e_2$ into $U_{N_2}$. There are $N$ distinct characters of $A$. $A$ is used to index the characters. For $a \in A$, the mapping $\psi(a)$ of $A$ into $U$ defined by

$$\langle b, \psi(a) \rangle = u_1^{a_1 b_1} u_2^{a_2 b_2}, \quad b \in A,$$

is a character of $A$, and the mapping $\psi : A \to A^*$ is an isomorphism of $A$ onto $A^*$.

In general, if

$$A = \langle N_1 \times \cdots \times N_R \rangle,$$

a typical element $a \in A$ will be written as

$$a = (a_1, \ldots, a_R), \quad a_r \in \langle N_r \rangle, \quad r = 1, 2, \ldots R - 1.$$

The mapping $\psi(a)$ of $A$ into $U$ defined by

$$\langle b, \psi(a) \rangle = u_1^{a_1 b_1} \cdots u_R^{a_R b_R}, \quad b \in A,$$

where $u_r = e^{2\pi i/N_r}, \quad r = 1, 2, \ldots R$, is a character of $A$, and the mapping

$$\psi : A \to A^*$$

is an isomorphism of $A$ onto $A^*$. Since every finite Abelian group can be written in this form (Corollary 3.1.4), the following theorem has been proved.

**Theorem 3.2.2** If $A$ is a finite Abelian group, then $A$ is isomorphic to its character group.

The isomorphism between $A$ and its character group is not canonical, since it depends on presentation and choice of isomorphisms between the cyclic group factors of the presentation. Observe that the isomorphism given above satisfies

$$\langle b, \psi(a) \rangle = \langle a, \psi(b) \rangle, \quad a, b \in A.$$

Such an isomorphism is called *symmetric*.

An isomorphism $\psi$ from $A$ onto $A^*$ defines a mapping

$$\Psi_\psi : A \times A \to \langle N \rangle$$

by the rule

$$\langle a, \psi(b) \rangle = e^{(2\pi i/N)\Psi_\psi(a,b)}, \quad a, b \in A.$$
\[\Psi_\psi\text{ is a } /N\text{-bilinear form on } A \times A:\]
\[\Psi_\psi(a + a', b) = \Psi_\psi(a, b) + \Psi_\psi(a', b),\]
\[\Psi_\psi(a, b + b') = \Psi_\psi(a, b) + \Psi_\psi(a, b'),\quad a, a', b, b' \in A.\]

Conversely, a \(/N\text{-bilinear form on } A \times A\) uniquely determines a homomorphism of \(A\) onto \(A^*\). A \(/N\text{-bilinear form is called nonsingular}\) if the corresponding homomorphism is an isomorphism. The following theorem summarizes these results.

**Theorem 3.2.3** The formula
\[\langle a, \psi(b) \rangle = e^{(2\pi i/N)\Psi_\psi(a, b)}, \quad a, b \in A,\]
determines a one-to-one correspondence between the set of all isomorphisms \(\psi\) for \(A\) onto \(A^*\) and the set of all nonsingular \(/N\text{-bilinear forms } \Psi_\psi\text{ of } A.\)

For later reference the following theorem will be useful.

**Theorem 3.2.4** If \(b \in A\), then
\[\sum_{a^* \in A^*} \langle b, a^* \rangle = \left\{\begin{array}{ll} N, & b = 0, \\ 0, & b \neq 0. \end{array}\right.\]

If \(a^* \in A^*\), then
\[\sum_{b \in A} \langle b, a^* \rangle = \left\{\begin{array}{ll} N, & a^* = 0, \\ 0, & a^* \neq 0. \end{array}\right.\]

**Proof:** Let \(A = \mathbb{Z}/N_1 \times \cdots \times \mathbb{Z}/N_R\) and
\[A^* = \{\chi_a : a \in A \text{ and } \langle b, \chi_a \rangle = u_1^{a_1 b_1} \cdots u_R^{a_R b_R}, \ b \in A\},\]
where \(u_r = e^{2\pi i/N_r}, \ r = 1, 2, \ldots R\). Now fix \(b \in A\).

\[\sum_{a^* \in A} \langle b, a^* \rangle = \sum_{a \in A} \langle b, \chi_a \rangle = \sum_{(a_1, \ldots, a_r) \in \mathbb{Z}/N_1 \times \cdots \times \mathbb{Z}/N_R} u_1^{a_1 b_1} \cdots u_R^{a_R b_R},\]
\[= \sum_{i_1 = 0}^{N_1} (u_1^{b_1})^{i_1} \cdots \sum_{i_R = 0}^{N_R} (u_R^{b_R})^{i_R}.\]

If \(b = 0 \Leftrightarrow (b_1, \ldots, b_n) = (0, \ldots, 0)\) then \(u_r^{b_r} = 1, \ r = 1, 2, \ldots R\) and the sum equals \(N_1 \cdots N_R = N.\)
If \(b \neq 0 \Leftrightarrow (b_1, \ldots, b_n) \neq (0, \ldots, 0)\) then \(u_r^{b_r}, \ r = 1, 2, \ldots R\) is a root of unity \(\neq 1.\) The sum of all powers of a root of unity \(\neq 1\) equals 0. Therefore the sum also equals 0. \qed
3.3 Duality

The isomorphism defined in the previous section induces isomorphisms on certain subgroups of $A$. These subgroups can then be used to reduce a Fourier transform to a set of lower order Fourier transforms.

Unless otherwise specified, fix an isomorphism $\psi$ from $A$ onto $A^*$.

Definition 3.3.1 (Dual of Subgroup) For a subgroup $B$ of $A$ the dual $B^\perp$ is the set defined by

$$B^\perp = \{ a \in A : \langle b, \psi(a) \rangle = 1, \text{ for all } b \in B \}. \quad (3.3)$$

Definition 3.3.2 (Quotient Group) For a subgroup $B$ of $A$ the quotient group $A/B$ is the set of all $B$-cosets

$$A/B = \{ a + B : a \in A \},$$

with Abelian group addition,

$$(a + B) + (a' + B) = (a + a') + B.$$ 

$B^\perp$ is the orthogonal complement of $B$ relative to the bilinear form $\Psi_\psi$,

$$B^\perp = \{ a \in A : \Psi_\psi(b,a) = 0, \text{ for all } b \in B \}. $$

Since $\psi$ is an isomorphism of $A$ onto $A^*$,

$$\psi(B^\perp) = \{ \psi(b^\perp) : b^\perp \in B^\perp \}$$

is the subgroup of all characters of $A$ that act trivially on $B$, meaning that for all $a^* \in \psi(B^\perp)$, and for all $a \in B$ it holds that $\langle a, a^* \rangle = 1$. 

A set of elements in $A$

$$a_0, a_1, \ldots, a_{L-1}$$

is called a complete system of coset representatives for $A/B$ if every $a \in A$ can be written uniquely as

$$a = a_l + b, \quad l \in \{0, 1, \ldots, L - 1\}, \quad b \in B.$$ 

A coset representation for $A/B$ can be found in the following way (Rofheart [20]):

1. Create an array whose first row contains the elements of the subgroup $B$
2. Select an element from $A$ that has not appeared in any previous row and add it to the elements of $B$ to create the next row. Continue this step until all elements have been exhausted. Each row of the resulting array is a coset. The set of the first elements (also called coset leaders) of each row form a complete set of coset representatives.

An isomorphism $\psi : A \to A^*$ induces the following isomorphisms, which play a central role in the design of DFT algorithms.

**Definition 3.3.3 (Induced Isomorphisms)** Define the two isomorphisms

$$
\psi_1 : B^\perp \to (A/B)^*
$$
$$
\psi_2 : A/B^\perp \to B^*
$$

by

$$
\langle a + B, \psi_1(b^\perp) \rangle = \langle a, \psi(b^\perp) \rangle, \quad a \in A, \ b^\perp \in B^\perp
$$
$$
\langle b, \psi_2(a + B^\perp) \rangle = \langle b, \psi(a) \rangle, \quad a \in A, \ b \in B.
$$

**Example 3.3.1 $(A = \mathbb{Z}/N, \ N = N_1 N_2)$** Define the isomorphism $\psi$ by

$$
\langle a, \psi(a') \rangle = e^{2\pi i a'/N}, \quad a, a' \in A,
$$

and the subgroup $B$ by

$$
B = N_1. \mathbb{Z}/N = \{0, N_1, \ldots, (N_2 - 1)N_1\}.
$$

Then the dual is given by

$$
B^\perp = \{a \in \mathbb{Z}/N : e^{2\pi i a b/N} = 1, \ \forall b \in N_1. \mathbb{Z}/N\}.
$$

It holds that $e^{2\pi i a b/N} = 1$ if and only if $N$ divides $a b$. Since $N_1$ divides $b$ for all $b \in N_1. \mathbb{Z}/N$ and $N = N_1 N_2$ it follows that $N_2$ has to divide $a$ and therefore

$$
B^\perp = N_2. \mathbb{Z}/N = \{0, N_2, \ldots, (N_1 - 1)N_2\}.
$$

A coset representation for the quotient group $A/B$ can be found by the above algorithm and is given by

$$
A/B = \{0, 1, \ldots, N_1 - 1\}.
$$

The isomorphisms $\psi_1$ and $\psi_2$ are given by

$$
\langle j, \psi_1(kN_2) \rangle = e^{2\pi i j k /N_1}, \quad j, k \in \{0, 1, \ldots, N_1\},
$$
$$
\langle j N_1, \psi_2(k) \rangle = e^{2\pi i j k /N_2}, \quad j, k \in \{0, 1, \ldots, N_2\}.
$$
3.4 The Chinese Remainder Theorem

The Chinese Remainder Theorem (CRT) is a major tool in algorithm design. It is the basis of the Good-Thomas FFT algorithm (Good [9]). It can be stated in several ways, but in its complete form, it is a statement about rings. The ring version, and especially its use of idempotents, will be important in implementation, where it provides uniformity.

Definition 3.4.1 (Primary Factorization) A primary factorization of an integer \( N \) is any factorization

\[
N = N_1 N_2 \cdots N_R
\]

into pairwise relatively prime integers.

First, a weak form of the CRT using Abelian group structure only will be given, which can be seen as a partial converse to Theorem 3.1.4.

Theorem 3.4.1 The direct product of cyclic groups having pairwise relative prime orders is a cyclic group.

Proof: Suppose

\[
A = /N_1 \times /N_2 \times \cdots \times /N_R, \quad N = N_1 N_2 \cdots N_R,
\]

where the integers \( N_r, r = 1, 2, \ldots, R \), are pairwise relatively prime. Claim that the element \( e = (1, 1, \ldots, 1) \in A \) has order \( N \) and thus generates \( A \). Observe that \( ke = 0 \) in \( A \) implies that \( N_r | k, r = 1, 2, \ldots, R \). Since the integers \( N_r, r = 1, 2, \ldots, R \), are pairwise relatively prime, it follows that \( N | k \), proving the claim and the theorem.

The preceding theorem states that \( /N \) and \( /N_1 \times /N_2 \times \cdots \times /N_R \) are isomorphic as groups, whenever \( N = N_1 N_2 \cdots N_R \) is a primary factorization. The complete statement of the CRT goes further by explicitly defining a ring isomorphism and its inverse.

Theorem 3.4.2 (Chinese Remainder Theorem) \( /N \) and

\[
/ /N_1 \times /N_2 \times \cdots \times /N_R
\]

are isomorphic as rings whenever \( N_1 N_2 \cdots N_R \) is a primary factorization of \( N \).

Rather than proofing the theorem, an explicit isomorphism and its inverse will be constructed. The construction makes use of idempotents, which will now be introduced.
Theorem 3.4.3 Let $N = N_1 N_2 \cdots N_R$ be a primary factorization of $N$. Then there exist uniquely determined elements

$$e_1, e_2, \ldots, e_R \in \mathbb{Z}/N$$

satisfying

$$e_r \equiv 1 \mod N_r, \quad (3.5)$$
$$e_r \equiv 0 \mod N_s, \quad r, s \in \{1, 2, \ldots, R\}, \quad r \neq s. \quad (3.6)$$

The set $\{e_1, e_2, \ldots, e_R\}$ is called the complete system of idempotents for the primary factorization $N = N_1 N_2 \cdots N_R$.

Proof: An explicit definition of $e_r$, $r = 1, 2, \ldots, R$ will be given.

$$e_r = \frac{N}{N_r} \left( \left( \frac{N}{N_r} \right)^{-1} \mod N_r \right), \quad r = 1, 2, \ldots, R. \quad (3.7)$$

Note that $N/N_r = N_1 \cdots N_{r-1} N_{r+1} \cdots N_R$ is coprime to $N_r$, as $\gcd(N_r, N_s) = 1$ for $r \neq s$, hence the inverse in (3.7) exists. One can check that (3.7) satisfies (3.5) and (3.6).

Three more properties of idempotents are given, which in fact characterize complete sets of idempotents (without proof).

Lemma 3.4.4 If $\{e_1, e_2, \ldots, e_R\}$ is a complete system of idempotents for the primary factorization $N = N_1 \times N_2 \times \cdots \times N_R$ then

$$e_r^2 \equiv e_r \mod N, \quad r \in \{1, 2, \ldots, R\}, \quad (3.8)$$
$$e_r e_s \equiv 0 \mod N, \quad r, s \in \{1, 2, \ldots, R\}, \quad r \neq s, \quad (3.9)$$
$$\sum_{r=1}^{R} e_r \equiv 1 \mod N. \quad (3.10)$$

Proof: Fix $r$, $r \in \{1, 2, \ldots, R\}$ throughout the proof. By (3.5) and (3.6),

$$N_r e_r - 1 \quad \text{and} \quad N_s e_r, \quad r \in \{1, 2, \ldots, R\}, \quad r \neq s. \quad (3.11)$$

From (3.11) it follows that

$$N_r e_r (e_r - 1), \quad \text{and} \quad N_s e_r (e_r - 1), \quad r \in \{1, 2, \ldots, R\}, \quad r \neq s.$$

Since $\gcd(N_1, N_2, \ldots, N_R) = 1$, and $e_r^2 - e_r = e_r (e_r - 1)$,

$$N = N_1 N_2 \cdots N_R | (e_r^2 - e_r),$$
proving (3.8). Fix \( s, s \in \{1, 2, \ldots, R\}; s \neq r \). It follows that \( N_r|e_s \) and from (3.11) that \( N_r|e_r \) for \( r' \neq r \). Therefore

\[
N = N_1N_2 \cdots N_R|e_re_s, \quad r \neq s,
\]

showing (3.9).

Finally, it follows from (3.11) that

\[
N_r|(e_1 + e_2 + \cdots + e_R - 1) \quad r \in \{1, 2, \ldots, R\}
\]

Again, since \( \gcd(N_1, N_2, \ldots, N_R) = 1 \), it holds that

\[
N = N_1N_2 \cdots N_R|(e_1 + e_2 + \cdots + e_R - 1)
\]

showing (3.10). \(\square\)

With all this preliminary remarks the isomorphism, which proofs the Chinese remainder theorem, can be constructed.

**Proof: (Chinese Remainder Theorem)** Let \( \{e_1, e_2, \ldots, e_R\} \) be a complete system of idempotents for the primary factorization \( N = N_1N_2 \cdots N_R \). Define the mapping

\[
\phi: \quad /N_1 \times /N_2 \times \cdots \times /N_R \quad \longrightarrow \quad /N
\]

\[
(a_1, a_2, \ldots, a_R) \quad \longmapsto \quad (a_1e_1 + a_2e_2 + \cdots + a_R e_R) \mod N
\]

It remains to show that \( \phi \) is an isomorphism. Take \( a = (a_1, \ldots, a_R), \ b = (b_1, \ldots b_R) \in /N_1 \times \cdots \times /N_R \). Straightforward verification shows that

\[
\phi(a + b) = \phi(a) + \phi(b) \mod N.
\]

Lemma 3.4.4 implies

\[
\phi(a \cdot b) = \phi(a)\phi(b) \mod N.
\]

By (3.8) and (3.9)

\[
\phi(a)\phi(b) \equiv a_1b_1e_1^2 + (a_1b_2 + a_2b_1)e_1e_2 + \cdots + a_R b_R e_R^2
\]

\[
\equiv a_1b_1e_1 + \cdots + a_R b_R e_R \mod N.
\]

Formula (3.10) implies

\[
\phi(1, \ldots, 1) = 1 \mod N,
\]

proving that \( \phi \) is a ring-homomorphism.

To prove that \( \phi \) is a bijection, take any \( k \in /N \) and observe that

\[
k \equiv ((k \mod N_1)e_1 + \cdots + (k \mod N_R)e_R) \mod N.
\]
Since \( N_1 \times \cdots \times N_R \) and \( N \) have the same number of elements, this proves the \( \phi \) is bijective, completing the proof of the CRT.

From the above proof, it can be seen that the inverse \( \phi^{-1} \) of \( \phi \) is given by

\[
\phi^{-1}(k) = (k \mod N_1, \ldots, k \mod N_R), \quad k \in N.
\]

This implies that every \( k \in N \) can be written uniquely as

\[
k \equiv k_1e_1 + \cdots + k_Re_R \mod N,
\]

where \( k_r \in N_r, \ r = 1, 2, \ldots, R \).

By the CRT, \( N = N_1 \times \cdots \times N_R \), where \( N = p_1^{\alpha_1} \cdots p_M^{\alpha_M} \) is the factorization of \( N \) into distinct prime powers. More generally, the following result holds.

**Theorem 3.4.5** Suppose \( A \) is a finite Abelian group of order \( N \), with

\[
N = p_1^{\alpha_1} \cdots p_M^{\alpha_M},
\]

the factorization of \( N \) into distinct prime powers. Then \( A \) is isomorphic to the direct product

\[
A_1 \times \cdots \times A_M,
\]

where \( A_m \) is a finite Abelian group of order \( p_m^{\alpha_m} \) and is isomorphic to a direct product

\[
/p_m^{\alpha_m(1)} \times \cdots \times p_m^{\alpha_m(R)}.
\]

**Proof:** By the fundamental theorem,

\[
A \cong N_1 \times \cdots \times N_R, \quad N = N_1 \cdots N_R.
\]

Write

\[
N_r = p_1^{\alpha_1(r)} \cdots p_M^{\alpha_M(r)}, \quad r = \{1, 2, \ldots, R\}, \quad \alpha_m(r) \geq 0.
\]

By the CRT,

\[
N_r = /p_1^{\alpha_1(r)} \times \cdots \times /p_M^{\alpha_M(r)}.
\]

Rearranging the factors completes the proof of the theorem.

The factorization (3.12) is uniquely determined by the condition on the orders. In fact, if \( B_m \) is the subset of all elements in \( A \) having order a power of the prime \( p_m \), then \( B_m \) is the subgroup of \( A \) isomorphic to \( A_m \), and

\[
A = B_1 \oplus \cdots \oplus B_M.
\]

This direct sum decomposition of \( A \) is called the primary factorization of \( A \) and the factors \( B_m \) are referred to as primary factors. Many results in theory and algorithm design can first be proved for primary factors and then lifted to the general case using the direct sum. The CRT admits generalization to rings other than \( N \). In particular, the CRT can be proved for the ring of polynomials over any field. (Lang [16])
3.5 The Vector Space \( L(A) \)

Denote the space of all complex-valued functions on a finite set \( X \) by \( L(X) \). Then \( L(X) \) is a vector space over \( \mathbb{C} \) under function addition and complex multiplication:

\[
(f + g)(x) = f(x) + g(x), \quad f, g \in L(X), \quad x \in X, \\
(\alpha f)(x) = \alpha (f(x)), \quad \alpha \in \mathbb{C}, \quad f \in L(X), \quad x \in X.
\]

\( L(X) \) is a finite-dimensional vector space having dimension the order of \( X \). A basis for \( L(X) \) can be given by the evaluation functions

\[
\{ e_x : x \in X \}
\]

(3.14)

defined by setting

\[
e_x(y) = \begin{cases} 1, & y = x, \\ 0, & y \neq x, \end{cases} \quad y \in X.
\]

The basis of evaluation functions is called a canonical basis, since it depends solely on the set \( X \). By abuse of language, the basis will often be denoted by \( X \) itself. In practice, the set \( X \) must be ordered for the set (3.14) to be a basis. The ordering on \( X \) defines an ordering on the canonical basis.

An inner product is defined on \( L(X) \) by the rule

\[
(f, g) = \sum_{x \in X} f(x) \bar{g}(x), \quad f, g \in L(X),
\]

where \( \bar{g} \) denotes the complex conjugate of \( g \). The basis \( X \) is orthonormal,

\[
(e_x, e_y) = \begin{cases} 1, & y = x, \\ 0, & y \neq x, \end{cases} \quad x, y \in X.
\]

**Theorem 3.5.1** For \( a^*, b^* \in A^* \), it holds that

\[
(a^*, b^*) = \begin{cases} N, & a^* = b^*, \\ 0, & a^* \neq b^*. \end{cases}
\]

**Proof:** Since by definition

\[
(a^*, b^*) = \sum_{a \in A} \langle a, a^* \rangle \langle a, b^* \rangle = \sum_{a \in A} \langle a, a^* - b^* \rangle,
\]

the theorem follows from Theorem 3.2.4. \( \square \)

Theorem 3.5.1 implies that the set \( N^{-1/2} A^* \) is orthonormal in \( L(A) \) and hence linearly independent, implying the next result.

**Corollary 3.5.2** The character group \( A^* \) of the finite Abelian group \( A \) is an orthogonal basis of \( L(A) \).
Chapter 4

Fourier Transforms of Finite Abelian Groups

The standard definition of the multidimensional discrete Fourier transform (MDFT) assumes a fixed coordinate system representation of the indexing set. In this chapter the MDFT will be defined and explored in a more abstract setting, one that removes the dependence on coordinates and solely references the additive Abelian group structure of the indexing set. This approach highlights the fundamental role played by the duality between periodization and decimation in MDFT algorithms design. This duality lies in the heart of all standard and recently discovered divide-and-conquer MDFT algorithms. Emphasizing the unity underlying these algorithms permits a deeper understanding of their differences and how these differences can be exploited in implementation. This is especially true in the design of massively parallel FFT algorithms. Algorithm design is reduced to relatively few basic principles without having to account for the details of specific coordinates (Tolimieri et al. [24]).

4.1 The Fourier Transform of $A$

Definition 4.1.1 (Fourier Transform of a Finite Abelian Group) Let $A$ be a finite Abelian group. The Fourier transform of $A$ is the mapping $F$ from $L(A)$ into $L(A^*)$ defined by the formula

$$F(f)(a^*) = \sum_{a \in A} f(a) \langle a, a^* \rangle, \quad f \in L(A), \quad a^* \in A^*.$$  

The mapping $F$ is clearly linear and depends solely on $A$. If it is necessary to express this dependence the notation $F_A$ is used instead of $F$. The definition is independent of the isomorphism $\psi$ between $A$ and $A^*$. To make things more familiar, it is possible to make the following definition.

Definition 4.1.2 (Fourier Transform Presentation) Suppose

$$\psi : A \to A^*$$

is any isomorphism of $A$ onto $A^*$. Assign to $\psi$ the linear transform

$$F_\psi : L(A) \to L(A)$$
defined by
\[(F \psi f)(a) = F(f)(\psi(a)), \quad f \in L(A), \quad a \in A.\]

\(F \psi\) is called the Fourier transform presentation of \(A\) corresponding to the isomorphism \(\psi\).

Usually, one assigns to the presentation \(A = \langle N_1 \times \cdots \times N_R \rangle\) the isomorphism
\[\psi : A \to A^*\]
defined by the rule
\[\langle b, \psi(a) \rangle = u_1^{a_1 b_1} \cdots u_R^{a_R b_R}, \quad a, b \in A,\]
where \(u_r = e^{2\pi i / N_r}, \quad r = 0, 1, \ldots, R\). Then
\[F \psi f(a) = F f(\psi(a))\]
\[= \sum_{b \in A} f(b) \langle b, \psi(a) \rangle\]
\[= \sum_{b_1 \in /N_1} \cdots \sum_{b_R \in /N_R} f(b_1, \ldots, b_R) u_1^{a_1 b_1} \cdots u_R^{a_R b_R},\]
which is the formula for the \(N_1 \times \cdots \times N_R\) \(R\)-dimensional Fourier transform.

Since the isomorphism \(\psi\) is not unique, different isomorphisms lead to different Fourier transform presentations. To make sense, these various presentations must be related. The next result states the relationship.

**Theorem 4.1.1** If \(\psi_1\) and \(\psi_2\) are isomorphisms of \(A\) onto \(A^*\), then
\[F \psi_1 f(a) = F \psi_2 f(b), \quad f \in L(A),\]
where
\[a = \psi_1^{-1} \psi_2 b, \quad a, b \in A.\]

**Proof:** By definition,
\[F \psi_1 f(a) = \sum_{c \in A} f(c) \langle c, \psi_1(a) \rangle\]
\[= \sum_{c \in A} f(c) \langle c, \psi_2 \psi_1^{-1} \psi_1(a) \rangle\]
\[= \sum_{c \in A} f(c) \langle c, \psi_2(b) \rangle\]
\[= F \psi_2 f(b),\]
where \(b = \psi_1^{-1} \psi_1(a)\). The theorem follows. \(\square\)
$F_{\psi_1}$ and $F_{\psi_2}$ differ by the automorphism $\psi_2^{-1}\psi_1$ acting as a permutation of $A$. Any algorithm computing $F_{\psi_1}$ can be used to compute $F_{\psi_2}$. Distinct presentations of $A$ result in MDFTs of varying dimensions and sizes, but the entire collection can be used by any algorithm that computes one.

Auslander et al. [2, 3] developed an algorithm to explicitly compute the isomorphisms $\psi_1$ and $\psi_2$. They were able to show that for a fixed number of points there are Cooley-Tukey FFT algorithms that work for all multidimensional DFTs with the same number of input points. Changing the dimension is achieved by relabeling the input and the output and changing the “twiddle factors”. They call such an algorithm a *dimensionless FFT*. A short overview of dimensionless FFTs will be given in Chapter 5.

### 4.2 The Induced Fourier Transform

Throughout the remainder of this chapter, unless otherwise specified, fix an isomorphism $\psi$ from $A$ onto $A^*$ and a subgroup $B$ of $A$ of order $M$ with $N = LM$. The isomorphisms

$$\psi_1 : B^\perp \to (A/B)^*, \quad \psi_2 : A/B^\perp \to B^*,$$

defined in the previous chapter induce linear isomorphisms

$$F_1 : L(A/B) \to L(B^\perp),$$
$$F_2 : L(B) \to L(A/B^\perp),$$

by

$$F_1 f = F_{A/B} f \circ \psi_1, \quad f \in F(A/B),$$
$$F_2 f = F_B f \circ \psi_2, \quad f \in L(B).$$

$F_{A/B}$ and $F_B$ denote the abstract Fourier transforms of $A/B$ and $B$. By abuse of language, $F_1$ and $F_2$ will be called the Fourier transform presentations corresponding to $\psi_1$ and $\psi_2$. The Fourier transform presentation $F_\psi$ can be built from these induced “lower-order” Fourier transforms $F_1$ and $F_2$. The designation lower-order Fourier transform is justified by the following results.

For simplicity of notation, set $f(a_l) = f(a_l + B)$.

**Theorem 4.2.1** If $f \in L(A/B)$ and $\{a_0, a_1, \ldots, a_{L-1}\}$, is a complete system of coset representatives for $A/B$, then

$$F_1 f(b^\perp) = \sum_{l=0}^{L-1} f(a_l) \langle a_l, \psi(b^\perp) \rangle, \quad b^\perp \in B^\perp.$$
4. Fourier Transforms of Finite Abelian Groups

Proof: Direct computation shows that,

\[ F_{1}f(b^\perp) = F_{A/B}f(\psi_1(b^\perp)) \]
\[ = \sum_{b \in A/B} f(b) \langle b, \psi_1(b^\perp) \rangle \]
\[ = \sum_{l=0}^{L-1} f(a_l + B) \langle a_l + B, \psi_1(b^\perp) \rangle \]
\[ = \sum_{l=0}^{L-1} f(a_l) \langle a_l, \psi(b^\perp) \rangle \]

\[ \square \]

In the notation of Example 3.3.1,

\[ F_{1}f(kN_2) = \sum_{j=0}^{N_1-1} f(j)e^{2\pi ijk/N_1}, \quad k = 0, 1, \ldots, N_1 - 1. \]

The right hand side is the \( N_1 \)-point finite Fourier transform of the values \( f(0), f(1), \ldots, f(N_1 - 1) \).

**Theorem 4.2.2** If \( f \in L(B) \) and \( \{ c_0, c_1, \ldots, c_{M-1} \} \), is a complete system of coset representatives for \( A/B \perp \), then

\[ F_2f(c_0) = \sum_{b \in B} f(b) \langle b, \psi(c_0) \rangle, \quad m = 0, 1, \ldots M - 1. \]

This theorem can be proven analogous to the one above.

In the notation of example 3.3.1,

\[ F_2f(k) = \sum_{j=0}^{N_2-1} f(jN_1)e^{2\pi ijk/N_2}, \quad k = 0, 1, \ldots N_2 - 1. \]

The right hand side is the \( N_2 \)-point finite Fourier transform of the values \( F(0), f(N_1), \ldots, f((N_2 - 1)N_1) \).

### 4.3 Periodic and Decimated Data

In this section, the previously defined induced Fourier transforms will be related. First, the following definitions are needed.
Definition 4.3.1 \((B\text{-}periodic \text{ Function})\) A function \(f \in L(A)\) is called \(B\)-periodic if
\[
f(a + b) = f(a), \quad a \in A, \quad b \in B.
\]

Definition 4.3.2 \((B\text{-}decimated \text{ Function})\) A function \(f \in L(A)\) is called \(B\)-decimated if \(f\) vanishes off of \(B\),
\[
f(b) = 0, \quad b \notin B.
\]

A \(B\)-periodic function is constant on the elements of a \(B\)-coset and defines a function \(f \in L(A/B)\) by
\[
f(a + B) = f(a), \quad a \in A.
\]

Conversely, a function \(f \in L(A/B)\) defines a \(B\)-periodic function \(f \in L(A)\) by the same formula. The space of \(B\)-periodic functions in \(L(A)\) can be identified with \(L(A/B)\) and the space of \(B\)-decimated functions in \(L(A)\) can be identified with \(L(B)\).

A \(B\)-periodic function \(f \in L(A)\) is completely described by \(L = N/M\) values. It is reasonable to expect that the computation of the Fourier transform \(Ff\) reflects the reduced-size input. The next result supports this idea and establishes the first part of the duality between periodization and decimation.

**Theorem 4.3.1** If \(f \in L(A)\) is \(B\)-periodic, then its Fourier transform \(F_{\psi}f\) is \(B^\perp\)-decimated and can be computed by the formula
\[
F_{\psi}f(b^\perp) = MF_1(f)(b^\perp), \quad b^\perp \in B^\perp.
\]

**Proof:** Suppose \(f\) is \(B\)-periodic. Then for \(c \in A\)
\[
(F_{\psi}f)(c) = \sum_{a \in A} f(a) \langle a, \psi(c) \rangle
\]
\[
= \sum_{l=0}^{L-1} \sum_{b \in B} f(a_l + b) \langle a_l + b, \psi(c) \rangle
\]
\[
= \sum_{l=0}^{L-1} f(a_l) \langle a_l, \psi(c) \rangle \sum_{b \in B} \langle b, \psi(c) \rangle.
\]

If \(c \notin B^\perp\), then \(\psi(c)\) is not the trivial character restricted to \(B\), and by Theorem 3.2.4
\[
\sum_{b \in B} \langle b, \psi(c) \rangle = 0, \quad c \notin B^\perp,
\]
implying that $F_\psi f$ is $B^\perp$-decimated.
If $c = b^\perp \in B^\perp$, then
\[ \sum_{b \in B} \langle b, \psi(b^\perp) \rangle = M, \]
and by Theorem 4.2.1
\[ (F_\psi f)(b^\perp) = M \sum_{i=0}^{L-1} f(a_i) \langle a_i, \psi(b^\perp) \rangle \]
\[ = M (F_1 f)(b^\perp), \]
completing the proof of the theorem. 

Fig. 4.1 shows this relationship.

\[ \begin{array}{cc}
\text{f \text{B-periodic}} & \text{f \text{\in L(A/B)}} \\
F_\psi & F_1 \\
\text{F_\psi f \text{B^\perp-decimated}} & \text{F_1 f \text{\in F(B^\perp)}}
\end{array} \]

**Figure 4.1:** $F_\psi f(b^\perp) = MF_1(f)(b^\perp), \quad b^\perp \in B^\perp.$

A partial converse is given by the next result.

**Theorem 4.3.2** If $f \in L(A)$ is $B$-decimated, then its Fourier transform $F_\psi f$ is $B^\perp$-periodic, which, viewed as a function in $L(A/B^\perp)$, can be computed by the formula
\[ F_\psi f(c) = F_2 f(c), \quad c \in A/B^\perp. \] (4.4)

**Proof:** Take $a \in A$ and $b^\perp \in B$. Assume that $f$ is $B$-decimated. Then
\[ (F_\psi f)(a + b^\perp) = \sum_{b \in B} f(b) \langle b, \psi(a + b^\perp) \rangle \]
\[ = \sum_{b \in B} f(b) \langle b, \psi(a) \rangle \langle b, \psi(b^\perp) \rangle \]

Since $\psi(b^\perp)$ acts trivially on $b$, it follows further that
\[ = \sum_{b \in B} f(b) \langle b, \psi(a) \rangle \]
\[ = (F_\psi f)(a), \]
showing that $F_\psi f$ is $B^\perp$-periodic. The rest of the theorem follows from Theorem 4.2.2. \qed
Fig. 4.2 shows this relationship.

The following example builds up on example 3.3.1.

**Example 4.3.1** Recall that $A = \mathbb{Z}/N_1 \times \mathbb{Z}/N_2$, where $N = N_1 N_2$ and that

$$
B = N_1, \quad N = \{0, N_1, \ldots, (N_2 - 1)N_1\}, \\
B^\perp = N_2, \quad N = \{0, N_2, \ldots, (N_1 - 1)N_2\},
$$

and a coset representation for $A/B$ is given by $\{0, 1, \ldots, N_1 - 1\}$.

If $f$ is $B$-periodic, it is completely described by its values

$$
f(0), f(1), \ldots, f(N_1 - 1).
$$

The Fourier transform $F_\psi f$ is $B^\perp$-decimated and vanishes except at the points

$$
0, N_2, \ldots, (N_1 - 1)N_2,
$$

where its values are given by

$$
F_\psi f(nN_2) = N_1 \sum_{m=0}^{N_1-1} f(m)e^{(2\pi i/n)mn}, \quad n = 0, 1, \ldots, N_1 - 1.
$$

### 4.4 Periodization and Decimation

**Definition 4.4.1 (B-periodization)** For $f \in L(A)$, define the $B$-periodization of $f$, $\text{Per}_B f \in L(A)$ by

$$
(\text{Per}_B f)(a) = \sum_{b \in B} f(a + b), \quad a \in A. \quad (4.5)
$$

**Definition 4.4.2 (B-decimation)** For $f \in L(A)$, define the $B$-decimation of $f$, $\text{Dec}_B f \in L(A)$ by

$$
(\text{Dec}_B f)(a) = \begin{cases} 
  f(a), & a \in B, \\
  0, & a \notin B. 
\end{cases} \quad (4.6)
$$

$\text{Per}_B$ is a linear homomorphism of $L(A)$ onto the space of $B$-periodic functions in $L(A)$ and $\text{Dec}_B$ is a linear homomorphism of $L(A)$ onto the space of $B$-decimated functions in $L(A)$. The following results relate $F_\psi$ with the periodization and decimation operations.
Theorem 4.4.1 $F_{\psi} \circ \text{Per}_B = M \text{Dec}_{B^\perp} \circ F_{\psi}$.

Proof: For $f \in L(A)$, $\text{Per}_B f$ is $B$-periodic and by Theorem 4.3.1 $F_{\psi}(\text{Per}_B f)$ is $B^\perp$-decimated. It follows that for $b^\perp \in B^\perp$,

$$F_{\psi}(\text{Per}_B f)(b^\perp) = \sum_{a \in A} \sum_{b \in B} f(a + b) \langle a, \psi(b^\perp) \rangle$$

$$= \sum_{a \in A} \sum_{b \in B} f(a) \langle a - b, \psi(b^\perp) \rangle$$

$$= \sum_{a \in A} f(a) \sum_{b \in B} \langle a, \psi(b^\perp) \rangle \langle b, \psi(b^\perp) \rangle.$$

Since $\psi(b^\perp)$ acts trivially on $b$ it follows that $\langle b, \psi(b^\perp) \rangle = 1$ and

$$F_{\psi}(\text{Per}_B f)(b^\perp) = M \sum_{a \in A} f(a) \langle a, \psi(b^\perp) \rangle$$

$$= M(F_{\psi} f)(b^\perp),$$

completing the proof of the theorem.

Combining Theorems 4.3.1 and 4.4.1, the $B^\perp$-decimation of $F_{\psi} f$ can be computed by a lower-order Fourier transform of the $B$-periodization of $f$:

$$F_{\psi} f(b^\perp) = F_1(\text{Per}_B f)(b^\perp), \quad b^\perp \in B^\perp. \quad (4.7)$$

This formula is illustrated in Fig. 4.3.

![Diagram](image)

Figure 4.3: $F_{\psi} f(b^\perp) = F_1(\text{Per}_B f)(b^\perp)$, $b^\perp \in B^\perp$.

A similar argument shows the dual statement, which is given in the following theorem.
Theorem 4.4.2 \( F_\psi \circ \text{Dec}_B = \frac{M}{N} \text{Per}_{B^\perp} \circ F_\psi \)

Combining Theorems 4.3.2 and 4.4.2, the \( B^\perp \)-periodization of \( F_\psi f \) can be computed by a lower order Fourier transform of the \( B \)-decimation of \( f \):

\[
\text{Per}_{B^\perp}(F_\psi f)(c) = \frac{N}{M} F_2(\text{Dec}_B f)(c), \quad c \in A/B^\perp. \tag{4.8}
\]

This formula is illustrated in Fig. 4.4.

\[f\]

\[\text{Dec}_B f\]

Dec\(_B f\) \(\in\) \(L(B)\)

\[F_2\]

\(F_2(\text{Dec}_B f)\) \(\in\) \(L(A/B^\perp)\)

\[F_\psi\]

\(F_\psi f\)

\[\text{Per}_{B^\perp}\]

\(\text{Per}_{B^\perp}(F_\psi f)\)

**Figure 4.4:** \(\text{Per}_{B^\perp}(F_\psi f)(c) = \frac{N}{M} F_2(\text{Dec}_B f)(c), \quad c \in A/B^\perp.\)
Chapter 5

Abstract Multidimensional FFT Algorithms

The abstract definition of the Fourier transform and the statement of Fourier transform duality as expressed by the periodization-decimation results of the previous chapter provide a unifying principle underlying most one-dimensional and multidimensional FFTs. In Section 5.1 it will be seen that the Good-Thomas FFT is an immediate consequence of the abstract definition when the indexing set can be written as the direct sum, \( A = B \oplus C \), of subgroups \( B \) and \( C \).

In general, however, it cannot be assumed that \( A \) can be written nontrivially as a direct sum of subgroups. For the Cooley-Tukey FFT, which will be discussed in Section 5.2, it is only required that \( A \) contains a nontrivial subgroup \( B \).

Finally, in Section 5.3, both algorithms will be used to show that it is possible to compute DFTs with the same number of input points, but of different dimensions with the same algorithm. Changing the dimension is achieved by relabeling the input and the output and changing the twiddle factors. Such an algorithm is called dimensionless FFT.

5.1 The Abstract Good-Thomas FFT

Consider a finite Abelian group \( A \) having the form of a group direct sum

\[
A = B \oplus C.
\]

A typical element \( a \in A \) can be written as \( a = (a, b) \), \( b \in B \) and \( c \in C \). Suppose that \( \psi_B \) and \( \psi_C \) are isomorphisms from \( B \) and \( C \) onto their character groups and set \( \psi = \psi_B \times \psi_C \). Observe that \( C = B^\perp \) with respect to \( \psi \).

For \( f \in L(A) \) and \( a' = (b', c') \in A \),

\[
(F_\psi f)(a') = \sum_{c \in C} \left( \sum_{b \in B} f(b, c) \langle b, \psi_B(b') \rangle \right) \langle c, \psi_C(c') \rangle.
\] (5.1)

Computing \( F_\psi \) proceeds through the following stages.

1. For each \( c \in C \), define the slices \( f_c \in L(B) \) by

\[
f_c = f(b, c), \quad b \in B.
\]
2. Compute the Fourier transforms of these slices
\[ F_{\psi_B} f(c'), \quad c' \in B. \]
3. For each \( b' \in B \) define \( g_{b'} \in L(C) \)
\[ g_{b'}(c) = F_{\psi_B} f(c'). \]
4. Compute the Fourier transforms
\[ F_{\psi_C} (g_{b'})(c). \]
5. Finally set
\[ F_{\psi} f(d') = F_{\psi_C} (g_{b'})(c'). \]

Note, that two data transpositions are required. One is step 3 and one in step 5. In many applications, a fixed isomorphism \( \chi \) of \( A \) onto \( A^* \) is given. In this case, Theorem 4.1.1 can be applied to compute \( F_{\chi} f \) by first computing \( F_{\psi} f \) followed by the appropriate array Permutation. The following theorem summarizes these results.

**Theorem 5.1.1** Let \( A \) be a finite Abelian group and \( B \) a subgroup of \( A \) such that there exists a subgroup \( C \) of \( A \) with \( A = B \oplus C \) and \( C \) isomorphic to \( A/B \). Further assume, that bases have been chosen for \( L(A) \) and \( L(A^*) \) so that \( F(A) \) can be written as a matrix. Then there exist permutation matrices \( P \) and \( Q \) such that
\[ F(A) = Q(F(B) \otimes I_{|C|})(I_{|B|} \otimes F(C))P = Q(F(B) \otimes F(C))P. \quad (5.2) \]

The best known case of the Good-Thomas DFT, sometimes called the prime factor FFT (Good [9]) is when \( A = \mathbb{Z}/N \), where \( N = PQ \), \( P \) and \( Q \) relatively prime and the task is to compute the one-dimensional \( N \)-point Fourier transform. By the CRT, the cyclic group \( \mathbb{Z}/N \) is isomorphic to the direct product of the cyclic groups \( \mathbb{Z}/P \) and \( \mathbb{Z}/Q \).

\[ \phi : \mathbb{Z}/N \cong \mathbb{Z}/P \times \mathbb{Z}/Q. \quad (5.3) \]

The isomorphism usually attached to \( \mathbb{Z}/N \),
\[ a \mapsto \psi_a(b) = e^{2\pi i(ab/N)}, \quad a, b \in \mathbb{Z}/N, \quad (5.4) \]
leads to the one-dimensional \( N \)-point Fourier transform, while the isomorphism usually attached to \( \mathbb{Z}/P \times \mathbb{Z}/Q \),
\[ (a, b) \mapsto \psi_{(a,b)}(c, d) = e^{2\pi i((ac/P)+(bd/Q))}, \quad a, c \in \mathbb{Z}/P, \quad b, d \in \mathbb{Z}/Q, \quad (5.5) \]
leads to the two-dimensional \( P \times Q \) Fourier transform. Up to data permutation, they are equivalent computations.

With the help of the Chinese remainder theorem it is possible to calculate the permutations \( P \) and \( Q \) in Theorem 5.1.1 for the one-dimensional case.
Definition 5.1.1 (Chinese Remainder Theorem Permutation) The Chinese remainder theorem implies that $/N \cong /N_1 \times \cdots /N_R$ when $\gcd(N_i, N_j) = 1$ for $i \neq j$. The isomorphism is given by the complete system of idempotents $\{e_1, e_2, \ldots, e_R\}$ for the primary factorization $N = N_1 N_2 \cdots N_R$ and defines a permutation of the integers $0, 1, \ldots, N - 1$.

$$\text{CRT}(N_1, \ldots, N_R)(e_i^{N_1} \otimes \cdots \otimes e_i^{N_R}) = e_i^{N_1 \mod N_1} \otimes \cdots \otimes e_i^{N_R \mod N_R}, \quad (5.6)$$

where $e_i^N$ is the vector of size $N$ with a 1 in the $i$th position and zeros elsewhere. Further, for $\gcd(N, a) = 1$ define $S_a^N$ to be the permutation matrix that maps $e_i^N$ to $e_{ai \mod N}^N$.

Theorem 5.1.2 (Prime Factor FFT) Let $N = N_1 N_2 \cdots N_R$ with $N_r = p_r^{a_r}$. Then

$$F(N) = Q(N_1, \ldots, N_R)(F(N_1) \otimes \cdots \otimes F(N_R))P(N_1, \ldots, N_R)$$

$$= Q(N_1, \ldots, N_R) \left( \prod_{r=1}^{R} I_{N(r-1)} \otimes F(N_r) \otimes I_{\overline{N}(r)} \right) P(N_1, \ldots, N_R), \quad (5.7)$$

where

$$P(N_1, \ldots, N_R) = \text{CRT}(N_1, \ldots, N_R),$$

$$Q(N_1, \ldots, N_R) = \text{CRT}(N_1, \ldots, N_R)^{-1}(S_{e_1 N_1}^{N_1} \otimes \cdots \otimes S_{e_r N_R}^{N_R}),$$

$$N(r - 1) = N_1 N_2 \cdots N_{r-1}, \quad N(0) = 1,$$

$$\overline{N}(r) = N/N(r).$$

Proof: van Loan [25]. □

5.2 The Abstract Cooley-Tukey FFT

The Good-Thomas FFT applies whenever $A$ can be written as a direct sum $A = B \oplus C$. More generally, every subgroup $B$ of a finite Abelian group $A$ determines an abstract Cooley-Tukey FFT even if $B$ no longer splits in $A$.

Consider a finite Abelian group $A$ and an isomorphism $\psi$ of $A$ onto $A^\ast$. Let $B$ be a subgroup of $A$, $B^\perp$ its and $C = A/B$.

Take coset representatives for $C = A/B$

$$a_0, a_1, \ldots, a_{L-1}, \quad a_0 = 0, \quad (5.8)$$

and coset representatives for $A/B^\perp$

$$c_0, c_1, \ldots, c_{M-1}, \quad c_0 = 0. \quad (5.9)$$
A straightforward proof of the multidimensional Cooley-Tukey FFT relative to the subgroup $B$ will be given. The interpretation of the stages will reveal the underlying periodization–decimation structure.

For $f \in L(A)$, $F_\psi(f)$ can be computed on the coset $c_m + B^\perp$ by

$$F_\psi f(c_m + b^\perp) = \sum_{l=0}^{L-1} \sum_{b \in B} f(a_l + b) \langle a_l + b, \psi(c_m + b^\perp) \rangle$$

$$= \sum_{l=0}^{L-1} \left[ \left( \sum_{b \in B} f(a_l + b) \langle b, \psi(c_m) \rangle \right) \langle a_l, \psi(b^\perp) \rangle \right],$$

with $b^\perp \in B^\perp$ and $m \in \{0, 1, \ldots, M - 1\}$. $F_\psi f$ can be computed in the following sequence of steps.

1. Form the $B$-decimated functions $f_l(b) \in L(B)$, $l = 0, 1, \ldots, L - 1,$

   $$f_l(b) = f(a_l + b), \quad b \in B.$$

2. Compute the Fourier transforms $F_2f_l \in L(A/B^\perp)$, $l = 0, 1, \ldots, L - 1$, where $F_2$ is the Fourier Transform of $L(B)$ onto $L(A/B^\perp)$ induced by $\psi$.

   $$F_2f_l(c_m) = \sum_{b \in B} f_l(b) \langle b, \psi(c_m) \rangle, \quad m = 0, 1, \ldots, M - 1.$$

3. Form the functions $g_m \in L(A/B)$, $m = 0, 1, \ldots, M - 1$,

   $$g_m(a_l) = F_2f_l(c_m), \quad l = 0, 1, \ldots, L - 1.$$

4. Compute the products $h_m \in L(A/B)$, $m = 0, 1, \ldots, M - 1$,

   $$h_m(a_l) = g_m(a_l) \langle a_l, \psi(c_m) \rangle, \quad l = 0, 1, \ldots, L - 1.$$

5. Compute the Fourier transforms $F_1h_m \in L(B)$, $m = 0, 1, \ldots, M - 1$,

   $$F_1h_m(b^\perp) = \sum_{l=0}^{L-1} h_m(a_l) \langle a_l, \psi(b^\perp) \rangle, \quad b^\perp \in B^\perp.$$

6. Form the transpose

   $$F_\psi f(c_m + b^\perp) = f_1h_m(b^\perp), \quad m = 0, 1, \ldots, M - 1, \quad b^\perp \in B^\perp.$$

The abstract Cooley-Tukey FFT can be summarized by the following theorem.

**Theorem 5.2.1** Let $A$ be a finite Abelian group, $B$ a subgroup of $A$, $C = A/B$ and assume that bases have been chosen for $L(A)$ and $L(A^*)$ so that $F(A)$ can be written as a matrix. Then

$$F(A) = Q(F(B) \otimes I_{|C|})T(I_{|B|} \otimes F(C))P,$$

where $A$ and $P$ are permutations and $T$ is a diagonal matrix.
5.3 The Dimensionless FFT

In this section the results of the previous two sections will be used to develop an algorithm that is able to compute all multidimensional DFTs with the same number of input points. Changing the dimension is achieved by relabeling the input and the output and changing the twiddle factors. Such an algorithm is called dimensionless FFT (Austrander et al. [4]). The dataflow of these algorithms are fixed, independent of dimension.

An important consequence of this result, is that a program designed to compute the 1-dimensional DFT can be easily modified to compute the 2-dimensional and 3-dimensional DFT on the same number of input points. For example, it is possible to design a chip that can compute both a 1024-point 1-dimensional DFT, as well as a 32 × 32-point 2-dimensional DFT.\(^1\)

**Example 5.3.1** The one-dimensional 16-point FFT can be computed by the following factorization, which is the standard one-dimensional Cooley-Tukey FFT (Theorem 2.3.2),

\[
F(16) = (F(2) \otimes I_8)T_4(I_2 \otimes F(2) \otimes I_4)T_3(I_4 \otimes F(2) \otimes I_4)T_2(I_8 \otimes F(2))T_1 P, \tag{5.12}
\]

where \( \omega = e^{2\pi i/16} \)

\[
T_1 = I_{16} = \text{diag}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
\]

\[
T_2 = I_4 \otimes T_2(4) = \text{diag}(1,1,\omega^4,1,1,1,1,1,\omega^4,1,1,1,\omega^4,1,1),
\]

\[
T_3 = I_2 \otimes T_4(8) = \text{diag}(1,1,1,\omega^2,\omega^4,1,1,1,1,\omega^2,\omega^4,1,1,\omega^6),
\]

\[
T_4 = T_8(16) = \text{diag}(1,1,1,1,1,1,1,1,\omega,\omega^2,\omega^4,\omega^6,\omega^8),
\]

and

\[
P = R_{16} = (I_4 \otimes P(4,2))(I_2 \otimes P(8,2))P(16,2)
\]

is the bit-reversal matrix.

The two-dimensional FFT is equivalent to the factorization

\[
F(4) \otimes F(4) = (F(2) \otimes I_8)T_4(I_2 \otimes F(2) \otimes I_4)T_3(I_4 \otimes F(2) \otimes I_4)T_2(I_8 \otimes F(2))T_1 P, \tag{5.13}
\]

where \( \omega = e^{2\pi i/16} \)

\[
T_1 = I_{16} = \text{diag}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
\]

\[
T_2 = I_4 \otimes T_2(4) = \text{diag}(1,1,1,\omega^4,1,1,1,1,\omega^4,1,1,1,\omega^4,1,1,1),
\]

\[
T_3 = I_{16} = \text{diag}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1),
\]

\[
T_4 = T_2(4) \otimes I_4 = \text{diag}(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,\omega^4,\omega^4,\omega^4,\omega^4),
\]

and

\[
P = P(4,2) \otimes P(4,2).
\]

\(^1\)Patent #US6003056
Equations (5.12) and (5.13) are identical, except for the values in the diagonal matrices $T_i$ and the permutation matrix $P$. The dimensionless algorithm generalizing these examples and is formulated in the following theorem.

**Theorem 5.3.1 (Dimensionless FFT)** Let $\mathcal{F}_N$ denote an arbitrary multidimensional DFT of size $N$, and assume that $N = N_1 N_2 \cdots N_R$. Then there exist diagonal matrices $T_r$, $r = 1, 2, \ldots, R$, multidimensional DFTs $\mathcal{F}_{N_r}$ of size $N_r$, and permutation matrices $P$ and $Q$ such that

$$\mathcal{F}_N = Q \left\{ \prod_{r=1}^{R} (I_{N(r-1)} \otimes \mathcal{F}_{N_r} \otimes I_{\tilde{N}(r)}) T_r \right\} P, \quad (5.14)$$

where $N(r-1) = N_1 N_2 \cdots N_{r-1}$, $N(0) = 1$, and $\tilde{N}(r) = N / N(r)$. In the special case where $N = p^R$ and $N_r = p$ for $r = 1, 2, \ldots, R$, where $p$ is a prime, all of the DFTs occurring in (5.14) are equal to $F(P)$.

This theorem can be proved by regarding $\mathcal{F}_N$ as a Fourier transform on a finite Abelian group $A$ of order $N$ and using Theorem 5.2.1. Theorem 5.3.1 is equivalent to the following theorem once presentations are chosen and the Fourier transforms of Abelian groups are replaced by multidimensional DFTs.

**Theorem 5.3.2 (Generic Abelian FFT)** Let $A$ be a finite Abelian group of order $N$ and let $F(A)$ denote the Fourier transform associated with $A$. $F(A)$ can be viewed as a matrix as soon bases are chosen for $L(A)$ and $L(A^*)$.

Suppose $N = N_1 N_2 \cdots N_R$, then

$$F(A) = Q_A \left\{ \prod_{r=1}^{R} (I_{N(r-1)} \otimes F(A_r) \otimes I_{\tilde{N}(r)}) T_r \right\} P_A, \quad (5.15)$$

where $N(r-1) = N_1 N_2 \cdots N_{r-1}$, $N(0) = 1$, and $\tilde{N}(r) = N / N(r)$, $T_i$ is a diagonal matrix, $Q_A$ and $P_A$ are permutation matrices and $A_r$ is an Abelian group of order $N_r$.

**Proof:** Auslander et al. [4]. \hfill $\square$

A generic version of the prime factor FFT can be obtained in a similar fashion using Theorem 5.1.1.

**Theorem 5.3.3 (Generic Prime Factor FFT)** Let $A$ be a finite Abelian group of order $N$ and let $F(A)$ denote the Fourier transform associated with $A$. $F(A)$ can be viewed as a matrix as soon bases are chosen for $L(A)$ and $L(A^*)$. 

Suppose \( N = N_1 N_2 \cdots N_R \), where \( N_r = p_r^{a_r} \), \( r = 1, 2, \ldots, R \) and \( p_1, p_2, \ldots, p_R \) are distinct primes, then

\[
F(A) = Q_A(F(A_1) \otimes F(A_2) \otimes \cdots \otimes F(A_R))P_A
\]

\[
= Q_A \left\{ \prod_{r=1}^{R} (I_N(r^{-1}) \otimes I_{\tilde{N}(r)} \otimes I_{T_i}) \right\} P_A,
\]

where \( N(r^{-1}) = N_1 N_2 \cdots N_{r-1} \), \( N(0) = 1 \), and \( \tilde{N}(r) = N/N(r) \), \( T_i \) is a diagonal matrix, \( Q_A \) and \( P_A \) are permutation matrices and \( A_r \) is a \( p_r \)-group of order \( N_r \).

**Proof:** Auslander et al. [4].

Theorems 5.3.2 and 5.3.3 can be proved by induction on \( R \) and by applying simple tensor product manipulation. However, these proofs are not constructive in the sense, that they don’t explicitly compute the twiddle factor matrices and the permutation matrices. Fortunately, constructive proofs exist.

Note that in Theorem 5.3.2 the factors \( N_1, N_2, \ldots, N_R \) can be chosen arbitrarily, whereas in Theorem 5.3.3 they have to be powers of distinct primes. This fact is also reflected in the constructive proofs of these theorems. The proof of Theorem 5.3.2 is rather simple and will be illustrated by an example below. The proof of Theorem 5.3.3 is far more complicated and would to go beyond the scope of this work.

**Example 5.3.2 (Dimensionless Prime Factor Algorithm)** Let \( F_N \) denote an arbitrary multidimensional DFT of size \( N = 2^4 \times 3^3 \times 5^2 = 10,800 \). There exists permutations \( P \) and \( Q \) such that

\[
F_N = P(F_{2^4} \otimes F_{3^3} \otimes F_{5^2})Q,
\]

where \( F(2^4) \), \( F(3^3) \), and \( F(5^2) \) are multidimensional DFTs of a given type of size \( 2^4 \), \( 3^3 \), and \( 5^2 \).

For example, consider

\[
F_N = F(30) \otimes F(18) \otimes F(20).
\]

Using the prime factor FFT (Theorem 5.1.2) on each of the three factors, it follows that

\[
F_N = Q(2,3,5)(F(2) \otimes F(3) \otimes F(5))P(2,3,5) \otimes Q(2,3^2)(F(2) \otimes F(3^2))P(2,3,5) \otimes Q(2^2,5)(F(2^2) \otimes F(5))P(2^2,5) = (Q(2,3,5) \otimes Q(2,3^2) \otimes Q(2^2,5)) \otimes ((F(2) \otimes F(3) \otimes F(5)) \otimes (F(2) \otimes F(3^2)) \otimes (F(2^2) \otimes F(5))) \otimes (P(2,3,5) \otimes P(2,3^2) \otimes P(2^2,5)).
\]

Using Theorem 2.2.2 to rearrange the tensor product of DFT matrices, it follows that

\[
F_N = (Q(2,3,5) \otimes Q(2,3^2) \otimes Q(2^2,5))L_{1-1}(2,2^2,3,3^2,5,5) \otimes ((F(2) \otimes F(2) \otimes F(2^2)) \otimes (F(3) \otimes F(3^2)) \otimes (F(5) \otimes F(5))) \otimes L_{1}(2,2^2,3,3^2,5,5)(P(2,3,5) \otimes P(2,3^2) \otimes P(2^2,5)).
\]
where $\sigma$ is the permutation $(1, 4, 6, 2, 5, 3, 7)$.

As a second example, consider $\mathcal{F}_N = F(120) \otimes F(190)$. Following the same procedure as above, it follows that

$$
\mathcal{F}_N = (Q(2^3, 3, 5) \otimes Q(2, 3^2, 5))L_{\sigma^{-1}}(2, 2^3, 3, 3^2, 5, 5) \\
((F(2) \otimes F(2^3)) \otimes (F(3) \otimes F(3^2)) \otimes (F(5) \otimes F(5))) \\
L_{\sigma}(2, 2^3, 3, 3^2, 5, 5)(P(2^3, 3, 5) \otimes P(2, 3^2, 5)),
$$

where $\sigma$ is the permutation $(2, 3, 5, 1, 4, 6)$.

The dimensionless prime factor algorithm is a generalization of this example.

**Theorem 5.3.4 (Dimensionless Prime Factor Algorithm)** Let $\mathcal{F}_N$ be an arbitrary multidimensional DFT of size $N = N_1 N_2 \cdots N_R$ with $N_r = n_r^{a_r}$. Then there exists permutations $P$ and $Q$ such that

$$
\mathcal{F}_N = Q(F(N_1) \otimes \cdots \otimes F(N_R))P = Q \left( \prod_{r=1}^{R} I_{N(r-1)} \otimes \mathcal{F}_{N_r} \otimes I_{S(r)} \right) P, \quad (5.17)
$$

where $\mathcal{F}_{N_r}$ is a multidimensional DFT of size $N_r$.

**Proof:** Assume that $\mathcal{F}_N = F(M_1) \otimes F(M_2) \otimes \cdots \otimes F(M_S)$ is an $S$-dimensional $M_1 \times M_2 \times \cdots \times M_S$ DFT where $N = M_1 M_2 \cdots M_S$. Apply the prime factor FFT to $F(M_s)$ for each $s$ and pull the permutations to the front and to the back. Next apply the appropriate tensor permutation to collect the DFTs for each prime. $\square$
Chapter 6

Lines and Planes on Abelian Groups and their Duals

In this chapter the geometry of finite Abelian groups will be introduced. This geometry provides powerful mathematical tools for designing new massively parallel algorithms. The geometric concepts underlying these algorithms with special and explicit emphasis on the central role of duality will be developed.

6.1 Lines

Definition 6.1.1 (Line) A line in a finite Abelian group $A$ is a maximal cyclic subgroup of $A$. If $a \in A$ generates a line, then denote the line generated by $a$ by $L(a)$.

This definition is analogous to the definition of a line in an Euclidean space, but some geometric intuition is lost. In general such lines can double back due to the periodicity of the group addition. Two distinct lines can intersect in more than one point.

In the following example, the Chinese Remainder Theorem (CRT) will be used to describe explicitly an isomorphism between two presentations of an Abelian group $A$.

Example 6.1.1 ($A = \mathbb{Z}_6 \times \mathbb{Z}_6$) By the CRT, $\mathbb{Z}_6 \times \mathbb{Z}_6$ is isomorphic to $\mathbb{Z}_2 \times \mathbb{Z}_3$. With idempotents $e_1 = 3$, $e_2 = 4$ every $a \in \mathbb{Z}_6$ can uniquely be written as

$$a = a_1 e_1 + a_2 e_2, \quad a_1 \in \mathbb{Z}_2, \quad a_2 \in \mathbb{Z}_3.$$

Then

$$A \cong (\mathbb{Z}_2 \times \mathbb{Z}_2) \times (\mathbb{Z}_3 \times \mathbb{Z}_3),$$

and $a, b \in \mathbb{Z}_6$ can uniquely written as

$$(a, b) = (a_1, b_1)(e_1, e_1) + (a_2, b_2)(e_2, e_2), \quad a_1, b_1 \in \mathbb{Z}_2, \quad a_2, b_2 \in \mathbb{Z}_3.$$

Set $e_1 = (e_1, e_1), e_2 = (e_2, e_2)$. The lines in $A$ are given by

$$Le_1 + Me_2,$$

where $L$ is a line in $\mathbb{Z}_2 \times \mathbb{Z}_2$ and $M$ is a line in $\mathbb{Z}_3 \times \mathbb{Z}_3$. 

51
The lines in \( /2 \times /2 \) are
\[
L_1 = \{(0,0),(1,0)\},
L_2 = \{(0,0),(1,1)\},
L_3 = \{(0,0),(0,1)\}.
\]

The lines in \( /3 \times /3 \) are
\[
M_1 = \{(0,0),(1,0),(2,0)\},
M_2 = \{(0,0),(1,1),(2,2)\},
M_3 = \{(0,0),(1,2),(2,1)\},
M_4 = \{(0,0),(0,1),(0,2)\}.
\]

There are 12 distinct lines in \( /6 \times /6 \) given by
\[ e_1L_j + e_2M_k, \quad j = 1, 2, 3 \quad k = 1, 2, 3, 4. \]

For example, the line
\[ e_1L_2 + e_2M_3 \]
consists of the points
\[ \{(0,0),(4,2),(2,4),(3,3),(1,5),(5,1)\}. \]

There is some substantial redundancy in this case, as some points can lie on as many as 4 distinct lines.

### 6.1.1 The Prime Case

**The Square Case**

Consider the finite Abelian group
\[
A = /p \times /p,
\]
where \( p \) is a prime. The order of every nonzero element in \( A \) is \( p \). Thus, every nonzero element \( a \in A \) generates a line \( L(a) \in A \).

Consider an arbitrary nonzero point
\[ a = (a_1,a_2) \in A. \]

There are two possibilities. First, if \( p \) does not divide \( a_1 \) (denoted by \( p \nmid a_1 \)), the \( a_1 \) is invertible mod \( p \), and \( a \) can be written as
\[ a = a_1(1,a_1^{-1}a_2), \]
where \( a_1^{-1} \) is the inverse of \( a_1 \) mod \( p \). It follows that
\[ L(a) = L(1,a_1^{-1}a_2), \]
and hence every point \( a \in A \) having \( p \nmid a_1 \) lies on one of the \( p \) lines
\[
L(1, j) \quad j = 0, 1, \ldots, p - 1.
\] (6.1)

The second possibility is that
\[
a = (0, a_2) = a_2(0, 1), \quad p \nmid a_2,
\]
which implies that \( a \) lies on the line
\[
L(0, 1).
\] (6.2)

From the theory of groups it follows that the order on any subgroup \( H \) of a group \( G \) must divide the order of \( G \). Since lines in \( A \) have prime order, the \( p + 1 \) lines described in (6.1) and (6.2) are distinct, and any two have trivial intersection. These results are summarized in the following theorem.

**Theorem 6.1.1** If \( A \) is a finite Abelian group of the form
\[
A = /p \times /p, \quad p \text{ prime},
\]
then there are \( p + 1 \) distinct lines in \( A \) given by
\[
L(1, j), \quad j = 0, 1, \ldots, p - 1,
\]
\[
L(0, 1).
\]

Every nonzero point \( a \in A \) lies on a unique line and any two lines have trivial intersection.

**The General Case**

Now suppose that
\[
A = \underbrace{/p \times \cdots \times /p}_{N}, \quad p \text{ prime}.
\]

Consider a nonzero point
\[
a = (a_1, a_2, \ldots, a_N) \in A.
\]

There is a smallest integer \( J \in \{1, 2, \ldots, N\} \), such that
\[
a_1 \equiv a_2 \equiv \cdots \equiv a_{J-1} \equiv 0 \mod p
\]
and \( a_J \) is invertible mod \( p \). Write
\[
a = a_J(0, \ldots, 0, a_J^{-1}a_{J+1}, \ldots, a_J^{-1}a_N) \mod p.
\]

Arguing as before, the following result holds.
Theorem 6.1.2 If $A$ is a finite Abelian group of the form

$$A = \frac{\mathbb{Z}_{p} \times \cdots \times \mathbb{Z}_{p}}{\mathbb{Z}_N}, \quad p \text{ prime},$$

then there are

$$\frac{p^N - 1}{p - 1} = 1 + p + \cdots + p^{N-1}$$

distinct lines in $A$ given by

$$L(1, j_2, \ldots, j_N), \quad j_2, \ldots, j_N \in \{0, 1, \ldots, p - 1\}, \quad (6.3)$$

$$L(0, j_3, \ldots, j_N), \quad j_3, \ldots, j_N \in \{0, 1, \ldots, p - 1\}, \quad (6.4)$$

$$\vdots$$

$$L(0, \ldots, 0, 1). \quad (6.5)$$

Every nonzero point $a \in A$ lies on a unique line.

6.1.2 The Power of Prime Case

The Square Case

Consider the finite Abelian group

$$A = \frac{\mathbb{Z}_{p^R} \times \cdots \times \mathbb{Z}_{p^R}}{\mathbb{Z}_p}, \quad p \text{ prime, } R > 1.$$ 

Each element $a \in A$ has an order of a power of $p$ with the maximum order $p^R$. A typical point $a \in A$ has the form

$$a = (a_1, a_2), \quad a_1, a_2 \in \frac{\mathbb{Z}_{p^R}}{\mathbb{Z}_p}.$$ 

The following theorem characterizes the order of elements $a \in A$ in terms of the greatest common divisor

$$\gcd(a_1, a_2, p^R).$$ 

Since $a_1$ and $a_2$ are defined modulo $p^R$, $\gcd(a_1, a_2, p^R)$ is well-defined.

Theorem 6.1.3 An element $a \in A$ has order $p^R$ if and only if

$$\gcd(a_1, a_2, p^R) = 1.$$ 

More generally, the order of an element $a$ is $p^{R-r}$ if and only if

$$\gcd(a_1, a_2, p^R) = p^r.$$
Proof: Consider a nonzero \( a \in A \) satisfying \( \gcd(a_1, a_2, p^R) = p^r \) with \( r \in \{0, 1, \ldots, R\} \). Then \( a = p^rb \), where \( p \nmid b_1 \) or \( p \nmid b_2 \). The element \( b \) has order \( p^R \). Since
\[
p^{R-r}a = (p^rb_1, p^Rb_2) \equiv (0, 0) \mod p^R,
\]
the element \( a \) has order \( p^{R-r} \), proving the theorem in one direction. Conversely, if \( a \) has order \( p^{R-r} \), then
\[
p^{R-r}a_1 \equiv 0 \mod p^R,
p^{R-r}a_2 \equiv 0 \mod p^R.
\]
It follows that \( p^r \mid a_1 \) and \( p^r \mid a_2 \) and \( a \) is the largest integer with this property, completing the proof of the theorem. \( \square \)

**Corollary 6.1.4** \( b \in A \) generates a line if and only if \( \gcd(b_1, b_2, p^R) = 1 \) and every point \( a \in A \) lies on a line.

Suppose \( a \in A \) has order \( p^R \). Then \( p \nmid a_1 \) or \( p \nmid a_2 \). If \( p \nmid a_1 \), then \( a_1 \) is invertible modulo \( p^R \), and it follows that
\[
a = a_1(1, a_1^{-1}a_2) \mod p^R.
\]
The point \( a \) lies on one of the lines
\[
L(1, j), \quad j = 0, 1, \ldots, p^R - 1. \tag{6.6}
\]
On the other hand, if \( p \mid a_1 \) and \( p \nmid a_2 \), then write
\[
a = a_2(a_2^{-1}a_1, 1) \mod p^R,
\]
and the point \( a \) lies on one of the lines
\[
L(pk, 1), \quad k = 0, 1, \ldots, p^{R-1} - 1. \tag{6.7}
\]
The lines of (6.6) and (6.7) are distinct, and every line in \( A \) is of this form, leading to the next result.

**Theorem 6.1.5** The finite Abelian group
\[
A = \mathbb{Z}/p^R \times \mathbb{Z}/p^{R-1}, \quad p \text{ prime, } R > 1,
\]
has
\[
p^R + p^{R-1}
\]
distinct lines given by
\[
L(1, j), \quad j = 0, 1, \ldots, p^R - 1,
\]
\[
L(pk, 1), \quad k = 0, 1, \ldots, p^{R-1} - 1.
\]
Distinct lines can have nontrivial intersections, which will be characterized in the following definition and theorem.

**Definition 6.1.2** For an integer \( j \), define \( \nu_p(j) \) to be the highest power of \( p \) dividing \( j \). \( \nu_p(j) \geq 0 \) with \( \nu_p(j) = 0 \) if and only if \( p \) does not divide \( j \).

**Theorem 6.1.6** For \( j, k \in \{0, 1, \ldots, p^R - 1\} \),

\[
L(1, j) \cap L(1, k) = \text{gp}(p^{R-r}, p^{R-r}j), \quad r = \nu_p(k - j). \tag{6.8}
\]

For \( j \in \{0, 1, \ldots, p^R - 1\}, k \in \{0, 1, \ldots, p^{R-1} - 1\} \),

\[
L(1, j) \cap L(pk, 1) = (0). \tag{6.9}
\]

For \( j, k \in \{0, 1, \ldots, p^{R-1} - 1\} \),

\[
L(pk, 1) \cap L(pj, 1) = \text{gp}(p^{R-r}k, p^{R-r-1}), \quad r = \nu_p(k - j). \tag{6.10}
\]

**Proof:** Only (6.8) will be proved. The other cases follow similarly. The intersection \( L(1, j) \cap L(1, k) \) contains the point

\[(p^{R-r}, p^{R-r}j) \equiv (p^{R-r}, p^{R-r}k) \mod p^R,\]

and hence the subgroup of order \( p^r \)

\[\text{gp}(p^{R-r}, p^{R-1}j),\]

Conversely, a typical point \( a \in L(1, j) \cap L(1, k) \) satisfies

\[a = (a_1, a_1j) \equiv (a_1, a_1k) \mod p^R,\]

which by the condition on \( r \) implies \( p^{R-r} \mid a_1 \). Write \( a_1 = b_1p^{R-r} \). Then

\[a = b_1(p^{R-r}, p^{R-r}j) \in \text{gp}(p^{R-r}, p^{R-r}j),\]

proving the claim. \( \square \)

**Example 6.1.2** Suppose \( R = 2 \). Then \( A = p^2 \times p^2 \) has \( p^2 + p \) distinct lines. \( p^2 \) lines are given by

\[
\begin{array}{c}
L(1, 0) \\
L(1, 1) \\
\vdots \\
L(1, p - 1)
\end{array}
\begin{array}{c}
L(2, p) \\
L(1, p + 1) \\
\vdots \\
L(1, 2p - 1)
\end{array}
\begin{array}{c}
\cdots \\
\cdots \\
\cdots \\
\cdots
\end{array}
\begin{array}{c}
L(1, (p - 1)p) \\
L(1, p^2 - p + 1) \\
\vdots \\
L(1, p^2, 1)
\end{array}
\]

Any two distinct lines in the same column have trivial intersection, while any two distinct lines in the \( j \)th row have the intersection \( \text{gp}(p, pj) \), \( j = 0, 1, \ldots, p - 1 \). The remaining \( p \) lines are given by

\[
\begin{array}{c}
L(0, 1) \\
L(p, 1) \\
\cdots \\
L((p - 1)p, 1)
\end{array}
\]

Any two distinct lines in this collection have the intersection \( \text{gp}(0, p) \).

Distinct lines in \( /p \times /p \) and \( /p^R \times /p^R \) are given in Tables 6.1 and 6.2.
<table>
<thead>
<tr>
<th>A</th>
<th>Number of lines</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>./p × ./p</td>
<td>p + 1</td>
<td>(L(1,j),\ 0 \leq j &lt; p)</td>
</tr>
<tr>
<td>(./p^R \times ./p^R)</td>
<td>(p^R + p^{R-1})</td>
<td>(L(1,j),\ 0 \leq j &lt; p)</td>
</tr>
<tr>
<td>(L(1,k)), (0 \leq k &lt; p^{R-1})</td>
<td>(L(pk,1),\ 0 \leq k &lt; p^{R-1})</td>
<td></td>
</tr>
</tbody>
</table>

Table 6.1: Number of lines.

<table>
<thead>
<tr>
<th>A</th>
<th>Intersection</th>
</tr>
</thead>
<tbody>
<tr>
<td>./p × ./p</td>
<td>Trivial intersection</td>
</tr>
<tr>
<td>(./p^R \times ./p^R)</td>
<td>(L(1,j) \cap L(1,k) = \text{gp}(p^{R-r},p^{R-rj}),)</td>
</tr>
<tr>
<td>(L(pk,1) \cap L(pj,1) = \text{gp}(p^{R-r},p^{R-r-1}),)</td>
<td>(L(1,j) \cap L(pk,1) = (0))</td>
</tr>
</tbody>
</table>

Table 6.2: Intersection of lines, \(j \equiv k \mod p^r\).

The General Case

The general prime power case

\[
A = \underbrace{./p^R \times \cdots \times ./p^R}_{N}, \quad p \text{ prime},
\]

can be analyzed by the same arguments that served in the case of two cyclic factors. Only the results will be described here.

A point \(b \in A\) generates a lines if and only if

\[
\gcd(b_1, \ldots, b_N, p^R) = 1.
\]

Every point \(a \in A\) lies on a line. In fact, if \(\gcd(a_1, \ldots, a_N, p^R) = p^r\) then \(a = p^rb\), where \(a\) generates a line. Organize the lines of \(A\) into the following distinct subcollections \(\mathcal{L}_M, \ M = 1, 2, \ldots, N:\)

\[
\mathcal{L}_1 = \{L(j) : j_1 = 1\}.
\]

For \(M = 2, 3, \ldots, N,\)

\[
\mathcal{L}_M = \{L(j) : j_M = 1 \text{ and } p \mid j_1, \ldots, j_{M-1}\}.
\]
The number of lines in $\mathcal{L}_M$, $M = 0, 1, \ldots, N$, is

$$(p^R)^{N-M}(p^{R-1})^{M-1}.$$ 

Summing over $M = 0, 1, \ldots, N$, the number of lines in $A$ is

$$(p^{R-1})^{N-1}\left(\frac{p^N-1}{p-1}\right).$$

Lines taken from different subcollections $\mathcal{L}_{M_1}$ and $\mathcal{L}_{M_2}$, $M_1 \neq M_2$, have trivial intersection, but two lines in $\mathcal{L}_M$ can intersect. Consider two lines in $\mathcal{L}_M$, $L(j)$ and $L(k)$. Set

$$r = \min\{\nu_p(j_l - k_l) : l = 0, 1, \ldots, N\}.$$ 

Then

$$L(j) \cap L(k) = \text{gp}(p^{R-r}j).$$

### 6.2 Duals of Lines

Consider duals of lines in the finite Abelian group

$$A = \mathbb{Z}/M \times \mathbb{Z}/M$$

(6.11)

relative to the standard bilinear form on $A$,

$$\Psi(a, b) = a_1b_1 + a_2b_2, \quad a, b \in A.$$ 

(6.12)

The matrix of $\Psi$ relative to the presentation (6.11) is $I_2$, the $2 \times 2$ identity matrix.

The mapping $j$ of $A$

$$J(a) = (-a_2, a_1)$$

is an automorphism of $A$ satisfying

$$\Psi(J(a), J(b)) = \Psi(a, b), \quad a, b \in A,$$

(6.13)

where

$$J = \begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}.$$ 

Condition (6.13) is equivalent to the matrix condition

$$J^T\Psi J = J^T I_2 J = I_2 = \Psi.$$ 

The automorphism $J$ maps lines onto lines by the rule

$$J(L(a)) = L(Ja).$$
Theorem 6.2.1  For any line $L$ in $A$, $L^\perp = J(L)$.

Proof: Since $\Psi$ is bilinear,

$$L^\perp = \{ a \in A : \Psi(x, a) = 0 \},$$

where $x$ generates the line $L$. From

$$\Psi(x, Jx) = x_1(-x_2) + x_1x_2 = 0$$

it follows that $J(L) \subset L^\perp$.

Conversely, if $a \in L^\perp$,

$$a_1x_1 + a_2x_2 \equiv 0 \mod M.$$ 

Since

$$\gcd(x_1, x_2, M) = 1,$$

it holds that

$$x_1y_1 - x_2y_2 \equiv 1 \mod M,$$

for some $y_1, y_2 \in \mathbb{Z}/M$. Setting

$$b_2 \equiv y_1a_2 + y_2a_1 \mod M,$$

results in

$$a_1 \equiv -x_2b_2, \quad a_2 \equiv x_1b_2 \mod M.$$ 

Thus $a = b_2(-x_2, x_1) \in J(L)$, completing the proof of the theorem. \hfill \Box

6.3 Hyperplanes

Consider the finite Abelian group

$$R = \mathbb{Z}/N_1 \times \cdots \times \mathbb{Z}/N_d,$$

where $d - k + 1$ dimensions are of equal and prime size. For simplicity of notation allow those dimensions to be contiguous and write

$$R = A \times B,$$

$$A = \mathbb{Z}/N_1 \times \cdots \times \mathbb{Z}/N_{k-1} \times \mathbb{Z}/p,$$

$$B = \mathbb{Z}/p^{d-k}, \quad p \text{ prime.}$$

A set of covering $k$-dimensional hyperplanes and their duals for for $R$ will be derived. The case where the array is of equal and power of prime size in at least $d - k + 1$ dimensions is dealt with in Rofheart [20].
6. Lines and Planes on Abelian Groups and their Duals

In a manner similar to the way $L(a)$ defines a line generated by the point $a \in A$, define the hyperplane generated by the points $a_1, \ldots, a_k \in A$ to be the subgroup

$$H(a_1, \ldots, a_k) = \{s_1a_1 + \cdots + s_ka_k : s_i \in \ IN_i \}. \quad (6.14)$$

The dimension of a subgroup of $A$ is the number of linearly independent points $a_i \in A$ required to generate it by the definition of (6.14).

For $a \in R$ define the $k$-dimensional subgroup $H(a)$ to be the closure of the vector $a$ with the $k-1$ standard basis vectors

$$\delta_i(j) = \begin{cases} 
1, & \text{if } i = j, \\
0, & \text{otherwise}. 
\end{cases} \quad (6.15)$$

$$H(a) = \{s_1\delta_1 + \cdots + s_{k-1}\delta_{k-1} + s_ka : s_i \in \ IN_i \}. \quad (6.16)$$

The following theorem gives a minimal set of $|A|$ point $k$-dimensional hyperplanes that cover $R$. It can be seen as an extension of Theorem 6.1.2. Let $\theta(j)$ denote the vector of $j$ zeros, i.e., $\theta(3) = (0,0,0)$.

**Theorem 6.3.1** If $R$ is a finite Abelian group of the form

$$R = A \times B,$$

$$A = \ IN_1 \times \cdots \times \ IN_{k-1} \times \ IN, \\
B = (\ IN)^{d-k}, \quad p \text{ prime},$$

then there are

$$p^{d-k+1} - 1 \\ p - 1$$

$k$-dimensional hyperplanes that cover $R$. They are given by

$$H_0 = H((\theta(k-1), 1, j_{k+1}, \ldots, j_d)), \quad j_i \in \ IN, \\
H_1 = H((\theta(k-1), 0, 1, j_{k+2}, \ldots, j_d)), \\
\vdots \\
H_{d-k} = H((\theta(k-1), 0, \ldots, 0, 1)).$$

**Proof:** Every $a = (a_1, \ldots, a_d) \in R$ can be written

$$a = a^1 + \cdots + a^{k-1} + a',$$

where $a^i$ is the vector whose $i$th component is $a_i$ and zero elsewhere, and $a'$ is the vector $a' = (\theta(k-1), a_k, \ldots, a_d)$.

By definition of $H(a)$, $a^i$ is in every hyperplane of the covering set for $i = 1, 2, \ldots, k-1$ and applying similar arguments as in the proof of Theorem 6.1.2 it can be shown that every $a'$ is in at least one hyperplane of the covering set. □
The \(k\)-dimensional hyperplanes of the covering set of Theorem 6.3.1 contain redundant points. The number of points in \(R\) is
\[
|R| = N_1 \cdots N_{k-1} p^{d-k+1}.
\]
The number of points in each \(k\)-dimensional hyperplane is
\[
|A| = N_1 \cdots N_{k-1} p,
\]
and the number of redundant points is
\[
|A| \left( \frac{p^{d-k} - 1}{p - 1} \right).
\]
The locations of the redundant points are given by the \((k - 1)\)-dimensional hyperplane
\[
H(\delta_1, \ldots, \delta_{k-1}) = \{(s_1, \ldots, s_{k-1}, \theta (d - k + 1)) : s_i \in \mathbb{N}_0 \}
\]
Now the duals of the \(k\)-dimensional hyperplanes of the covering set of Theorem 6.3.1 will be derived. Take
\[
H_0 = H(a) = H(\delta_1, \ldots, \delta_{k-1}, a),
\]
where \(a = (\theta (k - 1), 1, \hat{j}_{k+1}, \ldots, \hat{j}_d)\). Then
\[
H_0^\perp = \{ b \in R : \Psi(x, b) = 0, \ \forall x \in \{ \delta_1, \ldots, \delta_{k-1}, a \} \}
= \{ b \in R : b_1 = 0, \ldots, b_{k-1} = 0, \ \Psi(a, b) = 0 \},
\]
where \(\Psi\) is the standard bilinear form on \(R\). The term \(\Psi(a, b) = 0\) can be evaluated to
\[
\Psi(a, b) = 0 \Leftrightarrow b_k + b_{k+1} \hat{j}_{k+1} + \cdots + b_d \hat{j}_d = 0
\Leftrightarrow b_k = -(b_{k+1} \hat{j}_{k+1} + \cdots + b_d \hat{j}_d).
\]
Finally,
\[
H_0^\perp = \left\{ b_k \delta_k + \cdots b_d \delta_d : b_{k+1}, \ldots, b_d \in \{0, 1, \ldots, p - 1\}, \ b_k = - \sum_{i=1}^{d} b_{k+i} \hat{j}_{k+i} \right\}.
\]
The duals of \(H_1, H_2, \ldots, H_{d-k}\) can be computed similarly and are summarized in the following theorem.
Theorem 6.3.2. The duals of the set of $k$-dimensional hyperplanes given by Theorem 6.3.1 are given by

\[ H_0^\perp = \left\{ b_k \delta_k + \cdots + b_d \delta_d : b_{k+1}, \ldots, b_d \in \{0, 1, \ldots, p - 1\}, \quad b_k = -\sum_{i=1}^{d} b_{k+i} \right\}, \]

\[ H_1^\perp = \left\{ b_k \delta_k + \cdots + b_d \delta_d : b_K, \ldots, b_d \in \{0, 1, \ldots, p - 1\}, \quad b_{k+1} = -\sum_{i=2}^{d} b_{k+i} \right\}, \]

\[ \vdots \]

\[ H_{d-k}^\perp = \left\{ b_k \delta_k + \cdots + b_d \delta_d : b_k, b_{k+1}, \ldots, b_{d-1} \in \{0, 1, \ldots, p - 1\}, \quad b_d = 0 \right\}. \]
Chapter 7

Reduced Transform Algorithms

This chapter presents the main results of this work, the design and implementation of reduced transform algorithms (RTAs). RTAs tend to minimize interprocessor communication at the cost of more complex input-output operations. They are based on the geometry of finite Abelian groups and the Fourier transform duality between periodization and decimation (Tolimieri et al. [24]).

Gertner [7] and Vulis [26] independently discovered dual versions of RTAs. Both versions will be outlined in Section 7.1. The design of the algorithms presented in Section 7.2 will follow Gertner [7].

Gertner first developed an algorithm for the two-dimensional case. He showed that the number of distinct $N$-point DFTs needed to calculate the $N \times N$-point DFT is equal to the number of linear congruences spanning the $N \times N$ grid. Since these linear congruences can be interpreted as lines, the algorithm was called line algorithm.

Later Gertner and Tolimieri [8] and Rofheart [20] extended the algorithm to the multidimensional case and introduced hyperplanes—that are subgroups of lower dimension—instead of lines. In the multidimensional case there is a higher degree of freedom for choosing a covering set of subgroups for the indexing set and therefore it is not just one algorithm, but a whole class of algorithms. Since these algorithms reduce the problem of calculating a multidimensional Fourier transform to a set of lower-order Fourier transforms, they are called reduced transform algorithms. Note that these lower-order Fourier transforms are completely independent and hence efficient parallel computation is feasible.

Later on the initial results have been extended to a wider range of transform sizes (An et al. [1]; Tolimieri et al. [22, 23, 24]).

7.1 General Structure of RTAs

In its most general form RTAs decompose the computation of the DFT into a collection of induced DFTs taken over a set of subgroups covering the index set. Depending on the choice of the subgroups, the following two versions of RTAs can be stated.
7.1.1 The Standard RTA

The standard RTA (or just RTA) begins with a set of subgroups \( \mathcal{B} \), whose duals cover \( A \) and computes \( F_\psi f \) by

- forming the collection of periodizations
  \[ \text{Per}_B f \in L(A/B), \quad B \in \mathcal{B}, \]

- computing the collection of induced DFTs
  \[ F_1(\text{Per}_B f), \quad B \in \mathcal{B}. \]

This completes the computation since \( F_1(\text{Per}_B f) \) equals \( F_\psi f \) on \( B^\perp \) by equation (4.7) on page 41 and \( \mathcal{B} \) is a dual covering of \( A \).

7.1.2 The Dual RTA

The dual RTA begins with a set of subgroups \( \mathcal{B} \), which cover \( A \). For each \( a \in A \) define the integer valued function \( \mu \) on \( A \) by

\[ \mu(a) = \text{the number of subgroups in } \mathcal{B} \text{ containing } a. \]

Define the weighted decimations of \( f \) by

\[ \text{Dec}_B^\mu f(a) = \begin{cases} \frac{1}{\mu(a)} f(a), & a \in B, \\ 0, & \text{otherwise}. \end{cases} \]

Since \( \mathcal{B} \) covers \( A \),

\[ f = \sum_{B \in \mathcal{B}} \text{Dec}_B^\mu f, \]

\[ F_\psi f = \sum_{B \in \mathcal{B}} F_\psi \text{Dec}_B^\mu f, \]

and \( F_\psi f \) can be computed by

- forming the collection of decimations
  \[ \text{Dec}_B^\mu f \in L(B), \quad B \in \mathcal{B}, \]

- computing the collection of induced FT
  \[ F_2(\text{Dec}_B^\mu f), \quad B \in \mathcal{B}. \]

Redundant computation is a necessary part of RTAs.

In applications, say, to the \( M \)-dimensional DFT, the collection \( \mathcal{B} \) is usually taken such that duals are a covering set of \( K \)-dimensional \((K < M)\) planes through the origin. The dimension \( K \) is an important design parameter as it affects local granularity and global parallelism (Tolimieri et al. [22]).
7.2 Examples

The examples given in this section are using the standard RTA. For several cases of the structure of the indexing set \( A \), sets of lines and hyperplanes and their duals were presented in Chapter 6. These results will be used to design RTAs for the following cases.

- 2-D RTA for \( A = \mathbb{Z} / p \times \mathbb{Z} / p \), \( p \) prime;
- 2-D RTA for \( A = \mathbb{Z} / p^r \times \mathbb{Z} / p^r \), \( p \) prime;
- \( d \)-dimensional RTA for \( A = \mathbb{Z} / N_1 \times \cdots \times \mathbb{Z} / N_d \), where \( A \) is of equal and prime size in \( d - k + 1 \) dimensions.

In the first two cases the periodization can easily be implemented in terms of shift and stride permutations. These two cases will be discussed in detail. The third case is an extension of the first two, it will be discussed only briefly.

7.2.1 2-D RTA for \( A = \mathbb{Z} / p \times \mathbb{Z} / p \), \( p \) Prime

Information necessary for designing the 2-D RTA for a \( p \times p \) array is listed in Table 7.1.

| Decimating Line \( L_j = L(1,j) \) | Periodizing Line \( L_j^\perp = L(-j,1) \) | Complement \( (L_j^\perp)^c = L(1,0) \)
| --- | --- | --- |
| Decimating Line \( L_p = L(0,1) \) | Periodizing Line \( L_p^\perp = L(1,0) \) | Complement \( (L_p^\perp)^c = L(0,1) \)

*Table 7.1: Covering lines and their duals for \( A = \mathbb{Z} / p \times \mathbb{Z} / p \), \( j = 0,1,\ldots,p-1 \).*

First the \( p+1 \) periodizations \( g_j = \text{Per}_{L_j^\perp} f \), \( j = 0,1,\ldots,p-1 \) have to be computed. Remember that \( g_j \) is completely determined by its values on \( (L_j^\perp)^c \).

\[
g_j(a) = (\text{Per}_{L(-j,1)} f)(a) = \sum_{b \in L(-j,1)} f(a+b), \quad a \in L(1,0), \quad j = 0,1,\ldots,p-1, \quad (7.1)
\]

\[
g_p(a) = (\text{Per}_{L(1,0)} f)(a) = \sum_{b \in L(0,1)} f(a+b), \quad a \in L(0,1). \quad (7.2)
\]
For simplicity of notation, write \( g_j(k) = g_j(0, 0) \), \( j = 0, 1, \ldots, p - 1 \) and \( g_p(k) = g_p(0, k) \). Then

\[
g_j(k) = \sum_{l=0}^{p-1} f(k - lj, l), \quad j = 0, 1, \ldots, p - 1, \tag{7.3}
\]

\[
g_p(k) = \sum_{l=0}^{p-1} f(l, k). \tag{7.4}
\]

Then, the Fourier transform of \( f \) on the Lines \( L_j, \ j = 0, 1, \ldots, p \) can be computed by the equation \( \text{Dec}_{L_j} F_\psi f = F_1(\text{Per}_{L_j} f) \) (equation (4.7) on page 41).

\[
F_\psi(b_j) = F_1(g_j)(b_j) = \sum_{a \in L(1, 0)} g_j(a) \langle a, \psi(b_j) \rangle, \quad b_j \in L(1, j), \quad j = 0, 1, \ldots, p - 1, \tag{7.5}
\]

\[
F_\psi(b) = F_1(g_p)(b) = \sum_{a \in L(0, 1)} g_p(a) \langle a, \psi(b) \rangle, \quad b \in L(0, 1). \tag{7.6}
\]

\[
F_\psi(l, lj) = \sum_{k=0}^{p-1} g_j(k) e^{2\pi i kl/p}, \quad j = 0, 1, \ldots, p - 1 \tag{7.7}
\]

\[
F_\psi(0, l) = \sum_{k=0}^{p-1} g_p(k) e^{2\pi i kl/p}. \tag{7.8}
\]

**Example 7.2.1** Consider the case \( p = 3 \). Take \( f \in L(A) \) and order \( f \) lexicographically

\[
f = \begin{bmatrix}
f_{00} \\
f_{10} \\
f_{20} \\
f_{01} \\
f_{11} \\
f_{21} \\
f_{02} \\
f_{12} \\
f_{22}
\end{bmatrix}.
\]

Let \( S_3 \) be the 3-point cyclic shift matrix

\[
S_3 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}
\]

and \( P(9, 3) \) is the 9-point stride by 3 matrix. Further define

\[
Z_3 = I_3 \oplus S_3 \oplus S_3^2.
\]
7. Reduced Transform Algorithms

Direct computation shows that (7.3) and (7.4) can be written as

\[ g_0 = (I_3 \otimes 1_3^T)P(9,3)f, \]
\[ g_1 = (I_3 \otimes 1_3^T)P(9,3)Z_3f, \]
\[ g_2 = (I_3 \otimes 1_3^T)P(9,3)Z_3^2f, \]
\[ g_3 = (I_3 \otimes 1_3^T)I_9f. \]

(7.9)  (7.10)  (7.11)  (7.12)

\( g_1 \) will be computed explicitly for illustration purpose.

\[
P(9,3)Z_3f = P(9,3)(I_3 \oplus S_3 \oplus S_3^2)f =
\begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix} = P(9,3)
\begin{bmatrix}
f_{00} \\
f_{10} \\
f_{20} \\
f_{01} \\
f_{11} \\
f_{21} \\
f_{02} \\
f_{12} \\
f_{22}
\end{bmatrix} = P(9,3) =
\begin{bmatrix}
f_{00} + f_{11} + f_{22} \\
f_{12} \\
f_{11} + f_{02} \\
f_{22} + f_{21} + f_{12}
\end{bmatrix}.
\]

(7.13)

In matrix notation, (7.9) - (7.12) can be written as

\[
\begin{bmatrix}
g_0 \\
g_1 \\
g_2 \\
g_3
\end{bmatrix} = \begin{bmatrix}
(I_3 \otimes 1_3^T)P(9,3) \\
(I_3 \otimes 1_3^T)P(9,3)Z_3 \\
(I_3 \otimes 1_3^T)P(9,3)Z_3^2 \\
(I_3 \otimes 1_3^T)I_9
\end{bmatrix}f.
\]

(7.13)

Finally the 3-point Fourier transforms \( F(3) \) have to be computed on the lines \( L_0, \ldots , L_3 \),

\[
F_{L_j} = F(3)g_j, \quad j = 0, 1, 2, 3,
\]

(7.14)

where \( F_{L_j} \) is the vector formed from the values of \( F \phi f \) on the line \( L_j \). Again, (7.14) can be written in matrix notation as

\[
\begin{bmatrix}
F_{L_0} \\
F_{L_1} \\
F_{L_2} \\
F_{L_3}
\end{bmatrix} = (I_4 \otimes F(3)) \begin{bmatrix}
g_0 \\
g_1 \\
g_2 \\
g_3
\end{bmatrix} = (I_4 \otimes F(3))Tf.
\]

(7.15)
This algorithm is illustrated in Fig. 7.1, where it is easy to see that the computations are completely independent and hence easily parallelizable.

The general case \( M = p, \) \( p \) prime, follows in exactly the same way.

\[
g_j = (I_p \otimes 1_p^T)P(p^2, p)Z_p^j f, \quad j = 0, 1, \ldots, p. \tag{7.16}
\]

where \( Z_p = I_p \oplus S_p \oplus \cdots \oplus S_p^{p-1} \), with \( S_p \) the \( p \)-point cyclic shift matrix. The output of each permutation is subjected to the same computational stage—a \( p \)-point Fourier transform \( F(p) \)—with the result

\[
F_{Lj} = F(p)g_j, \quad j = 0, 1, \ldots, p. \tag{7.17}
\]

Again, setting

\[
T = \begin{bmatrix}
(I_p \otimes 1_p^T)P(p^2, p) \\
(I_p \otimes 1_p^T)P(p^2, p)Z_p \\
\vdots \\
(I_p \otimes 1_p^T)P(p^2, p)Z_p^{p-1} \\
(I_p \otimes 1_p^T)I_p^2
\end{bmatrix} \tag{7.18}
\]

(7.16) and (7.17) can be written in matrix notation as

\[
\begin{bmatrix}
F_{L0} \\
F_{L1} \\
\vdots \\
F_{Lp}
\end{bmatrix} = (I_p+1 \otimes F(p)) \begin{bmatrix}
g_0 \\
g_1 \\
\vdots \\
g_p
\end{bmatrix} = (I_p+1 \otimes F(p))Tf. \tag{7.19}
\]
Now that the Fourier transform on the lines $L_j, j = 0, 1, \ldots, p$ have been computed, they have to be re-transformed in the array $F_\psi$. Unfortunately the transformation matrix can not be written in terms of Kronecker products and stride permutations. Moreover the re-transformation matrix is not unique, since all lines have the point $(0,0)$ in common. The matrix can be constructed as follows.

Order $F_\psi$ lexicographically and let $F_L$ denote the vector formed by concatenating all the vectors $F_{L_j}, j = 0, 1, \ldots, p$, to one big vector. Then the task is to find a matrix $T^* \in \{0, 1\}^{p \times p+1}$, such that

$$F_\psi = T^* F_L.$$  \hfill (7.20)

$T^*$ can be constructed row by row. For every row $m, m = 0, 1, \ldots, p^2$, find $k$ and $l$, $k, l \in \{0, 1, \ldots, p-1\}$, such that $m = k \cdot p + l$. Then find $i$ and $j$, $i \in \{0, 1, \ldots, p-1\}, j \in \{0, 1, \ldots, p\}$, such that $(k, l) = L_{ij}(i)$, where $L_{ij}(i)$ is the $i$th coordinate of the line $j$ and set $n = p \cdot j + i$. Row $m$ consists of all zeros except in column $n$, where it is one.

**Example 7.2.2** Again, consider the case $p = 3$. Equation (7.20) becomes

$$\begin{bmatrix}
F_{00} \\
F_{10} \\
F_{20} \\
F_{01} \\
F_{11} \\
F_{21} \\
F_{02} \\
F_{12} \\
F_{22}
\end{bmatrix} =
\begin{bmatrix}
1 & 1 & 1 \\
0 & 1 & 1 \\
1 & 0 & 1 \\
1 & 1 & 0 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1
\end{bmatrix}
\begin{bmatrix}
F_{00} \\
F_{01} \\
F_{02} \\
F_{10} \\
F_{11} \\
F_{12} \\
F_{20} \\
F_{21} \\
F_{22}
\end{bmatrix}$$

The results can be summarized in the following theorem.

**Theorem 7.2.1 (Reduced Transform Algorithm for $A = \mathbb{Z}/p \times \mathbb{Z}/p$)**

Let $A = \mathbb{Z}/p \times \mathbb{Z}/p$, $p$ prime and $f \in L(A)$. Order $f$ and $F_\psi f$ lexicographically and construct $T$ and $T^*$ as described above. Then the Fourier transform of $f$ can be computed by

$$F_\psi = T^* (I_{p+1} \otimes F(p)) T f,$$

where $F(p)$ is the $p$-point Fourier transform.

**7.2.2 2-D RTA for $A = \mathbb{Z}/p^r \times \mathbb{Z}/p^r$, $p$ Prime**

Information necessary for designing the 2-D RTA for a $p^r \times p^r$ array is listed in Table 7.2.
Table 7.2: Covering lines and their duals for $A = . /p^r \times . /p^r$, $j = 0, 1, \ldots, p^r - 1, k = 0, 1, \ldots, p^{r-1} - 1$.

The number of lines covering $A$ is $p^r + p^{r-1}$. In the first step the periodizations $\text{Per}_{L_j} f$, $j = 0, 1, \ldots, p^r + p^{r-1} - 1$ have to be computed.

\[
g_j(m) = \sum_{l=0}^{p^r-1} f(m - lj, l), \quad j = 0, 1, \ldots, p^r - 1 \tag{7.21}
\]

\[
g_{p^r+k}(m) = \sum_{l=0}^{p^{r-1}-1} f(l, m - pk), \quad k = 0, 1, \ldots, p^{r-1} - 1. \tag{7.22}
\]

In the second step the Fourier transforms of $g_j$, $j = 0, 1, \ldots, p^r + p^{r-1} - 1$ are computed and the results are stored back to the array $F_\psi f$.

\[
F_\psi(l, j) = \sum_{m=0}^{p^r-1} g_j(m) e^{2\pi i ml/p^r}, \quad j = 0, 1, \ldots, p^r - 1 \tag{7.23}
\]

\[
F_\psi(plt, l) = \sum_{m=0}^{p^{r-1}-1} g_{p^r+k}(m) e^{2\pi i ml/p^r}, \quad k = 0, 1, \ldots, p^{r-1} - 1. \tag{7.24}
\]

It is possible to derive the algorithm in matrix notation for the case $A = . /p^r \times . /p^r$. However, only the results here will be presented here. A detailed derivation can be found in Tolimieri et al. [24].

Set

\[
P_j = P(p^{2r}, p^r)Z_{p^r}^j, \quad j = 0, 1, \ldots, p^r,
\]

\[
Q_k = P(p^{2r}, p^r)W_{p^r}^k P(p^{2r}, p^r), \quad k = 0, 1, \ldots, p^{r-1},
\]

\[
Z_{p^r} = \bigoplus_{u=0}^{p^r-1} S_{p^r}^u,
\]

\[
W_{p^r} = I_p \otimes Z_{p^r}^p
\]

with $S_{p^r}$ the $p^r \times p^r$ cyclic shift matrix.
Then
\[
g_j = (I_{p^r} \otimes 1_{p^r}^T)f, \quad j = 0, 1, \ldots, p^r - 1, \\
g_{p^r + k} = (I_{p^r} \otimes 1_{p^r}^T)Q_k f, \quad k = 0, 1, \ldots, p^r - 1 - 1.
\]

The periodizations \(g_j, \ j = 0, 1, \ldots, p^r + p^r - 1 - 1\) are subjected to a \(p^r\)-point Fourier transform \(F(p^r)\)
\[
F_{L_j} = F(p^r)g_j, \quad j = 0, 1, \ldots, p^r + p^r - 1 - 1.
\]

Figure 7.2 summarizes the algorithm.

\[\begin{array}{c}
\text{Figure 7.2: Stages of the computation of the 2-D \(p^r \times p^r\) RTA.}
\end{array}\]

As in the previous section it is possible to find matrices \(T\) and \(T^*\), so that
\[
F_{\psi} = T^* (I_{p^r} \otimes p^{-1}) \otimes F(p^r) Tf
\]
holds. \(T\) is simply
\[
T = \begin{bmatrix}
(I_{p^r} \otimes 1_{p^r}^T)P_0 \\
\vdots \\
(I_{p^r} \otimes 1_{p^r}^T)P_{p^r - 1} \\
(I_{p^r} \otimes 1_{p^r}^T)Q_0 \\
\vdots \\
(I_{p^r} \otimes 1_{p^r}^T)Q_{p^r - 1 - 1}
\end{bmatrix},
\]
and \(T^*\) can be constructed similar to the method described in the previous section.
7.2.3 $d$-dimensional RTA

In this section a reduced transform algorithm for a group of the form

$$ R = A \times B $$

$$ A = . /N_1 \times \cdots \times . /N_{k-1} \times . /p $$

$$ B = ( . /p )^{d-k}, \quad p \text{ prime} $$

will be derived. This is a natural extension to the group considered in Section 7.2.1 and was introduced by Rofheart [20]. In Section 6.3 a set of covering hyperplanes and their duals for $R$ were described.

The $k$-dimensional hyperplanes $H_j$, $j = 0, 1, \ldots, d - k$ given by Theorem 6.3.1 and the $(d - k)$-dimensional duals of the hyperplanes $H_j^\perp$, $j = 0, 1, \ldots, d - k$ given by Theorem 6.3.2 can be defined by the homomorphisms

$$ \Phi_j : A \longrightarrow R, \quad \Phi_j(s) = (s_1, \ldots, s_{k-1}, \theta(j), s_k, s_kj_{j+1}, \ldots s_kj_{d-k}), \quad (7.25) $$

$$ \Phi_j'(b) = (\theta(k - 1), b_1, \ldots, b_{j-1}, - \sum_{t=j+1}^d b_tj_t, b_{j+1}, \ldots, b_{d-k}). \quad (7.26) $$

in the sense that $H_j = \Phi_j(A)$ and $H_j^\perp = \Phi_j'(B)$, $j = 0, 1, \ldots, d - k$. Further $A$ can be used as a coset representation for $R/H_j^\perp$.

In the first step of the algorithm the periodizations $g_j = \text{Per}_{H_j^\perp}$, $j = 0, 1, \ldots, d - k$ are computed. For $a \in R/H_j^\perp$,

$$ g_j(a) = (\text{Per}_{H_j^\perp})(a) = \sum_{b \in H_j^\perp} f(a + b) = \sum_{b \in B} f(a + \Phi_j'(b)). \quad (7.29) $$

$$ g_0(a) = \sum_{b \in B} f(a_1, \ldots, a_{k-1}, a_k - \sum_{t=1}^{d-k} b_tj_t, b_1, \ldots, b_{d-k}), \quad a \in A, $$

$$ g_1(a) = \sum_{b \in B} f(a_1, \ldots, a_{k-1}, b_1, a_k - \sum_{t=2}^{d-k} b_tj_t, i_2, \ldots, b_{d-k}), \quad a \in A, $$

$$ \vdots $$

$$ g_{d-k}(a) = \sum_{b \in B} f(a_1, \ldots, a_{k-1}, b_1, \ldots, b_{d-k}, a_k), \quad a \in A. $$
In the second step the $k$-dimensional Fourier Transforms are computed.

\[
F_\psi(b) = F_1(g_j)(b) = \sum_{a \in \mathbb{R}/H_j^\perp} g_j(a)\langle a, \psi(b) \rangle, \quad b \in H_j, \quad j = 0, 1, \ldots, d - k. 
\]

(7.30)

\[
F_\psi(\Phi_j(s)) = \sum_{a \in A} g_j(a)\omega_{N_1}^{s_1a_1} \cdots \omega_{N_k}^{s_ka_k}, \quad s \in A, \quad j = 0, 1, \ldots, d - k,
\]

where $\omega_{N_i} = e^{2\pi i / N_i}$. This completes the computation of the $d$-dimensional $\mathbb{R}/N_1 \times \cdots \times \mathbb{R}/N_k \times (\mathbb{R}/p)^{d-k}$ Fourier transform.
Chapter 8

Implementation of FFTs on Parallel Architectures

In this chapter, some issues concerning the parallel implementation of RTAs and FFT algorithms in general will be discussed.

8.1 Dataflow in Parallel FFTs

In parallel environments the cost of computing an FFT is dominated by data flow—memory access patterns—rather than arithmetic. For example, if all of the arithmetic is removed from a parallel FFT, i.e., floating point operations are replaced by NOPs, the resulting time is only reduced by approximately 10–25% (the actual percentage depends on the transform size) (Johnson et al. [14]).

On a workstation, the memory hierarchy consists of registers, primary cache, secondary cache, memory, and secondary storage, and the cost of a memory access depends on where in the memory hierarchy the data is located. On a distributed memory computer, there is an additional level in the memory hierarchy, namely, on-processor or local memory versus off-processor or global memory. In this situation, moving the data across the memory hierarchy involves network communication, and can be very costly compared to operations performed locally on a given processor. In experiments carried out on a CRAY T3E, it was observed that over 50% of the cost of a row-column 3-D FFT was due to global transpositions between the computation stages (Johnson and Johnson [13]).

Therefore it is necessary to decompose FFT algorithms into a communication layer and into a computation layer. To find an FFT algorithm suited best for a specific machine one first has to optimize the communication pattern to the actual machine and then find optimal computing kernels w.r.t. the computing nodes and the communication pattern.

In this sense, Auslander et al. [4] developed a special FFT algorithm, called dimensionless FFT, whose dataflow is independent of dimension. Changing the dimension is achieved by relabeling the input and the output and changing the twiddle factors. This suggests that an effective means of implementing and optimizing parallel FFT algorithms is to start with an “optimal” dataflow and then instantiate the dataflow, by providing appropriate twiddle factors and data relabelings.
The RTA also joins the class of algorithms, where dataflow and computation can be decomposed. In dimensions higher than two, it is possible to vary the granularity and the degree of parallelism of the algorithm by specifying the dimension of the hyperplanes in the output index set on which the DFT is computed. In the ideal case the number of hyperplanes is equal to the degree of parallelism of the machine. If this is not the case, there are additional tools for scaling by introducing a hybrid algorithm that nests the RTA within the multidimensional FFT.

### 8.2 Machine Model

In the following sections, parallel implementations of the row-column algorithm, the reduced transform algorithm and hybrid algorithms will be described. For a machine model consider a broadcast mode multiprocessor machine. Such machines typically feature a collection of homogeneous processing elements (nodes) together with an interconnection network of a regular topology for interprocessor communication. The node processors are externally connected to an I/O channel. The machine must support two communications functions, broadcast and report.

**Broadcast.** This function downloads data from the I/O channel to all processing elements (PE).

**Report.** This function allows a distinguished PE to upload the data to the I/O channel.

This general model covers multiprocessor architectures from buses and trees to hypercubes.

### 8.3 Parallel RC Algorithms

The multiprocessor implementation of the row-column (RC) FFT algorithm can be described as follows.

- Broadcast rows of the input multidimensional data uniformly among the processing elements. If there are more rows than processing elements, then several rows may be assigned to a particular processing element.
- In parallel, compute the DFT of the rows of the data in each of the processing elements.
- Globally transpose the intermediate results using the interconnection network of the machine.
In parallel, compute the DFT of the data in each of the processing elements.

Upload results. The transformed data is stored columnwise in the machine. If the data is required in its natural order, then an additional transpose must be carried out at some point of the uploading process.

For two-dimensional arrays, two global transposes are required to implement the row-column FFT. An additional global transpose is required for each increase in dimension. Implementation of global transpose on a parallel machine depends strongly on the interconnection network and topology. A great deal of research has been carried out for optimal implementation of global transpose specific to particular topologies, but they are still far from optimal (Tolimieri et al. [24]).

Assuming a degree of parallelism $N$, the relative performance (speed-up) of an $N \times N$ DFT is

$$\text{speed-up} = \frac{2N \text{FFT}(N)}{2 \text{FFT}(N)} = N.$$

### 8.4 Parallel RTAs

The RTA computes the $R$-dimensional DFT by reducing the $R$-dimensional dataset to a covering set of $S$-dimensional hyperplanes ($S < R$). The reduction is realized by a periodization operation that involves only additions and can be computed simultaneously with the data download. Assign to each node the computation of the DFT on one of the $S$-dimensional hyperplanes. Optimal parallel implementation occurs when the number of $S$-dimensional hyperplanes matches the degree of parallelism of the target machine. The computation can be divided into three stages.

- A global dimension reduction stage that downloads and simultaneously periodizes the data to the node processors. Since there is no data interdependency, this stage can proceed in parallel across the node processors. The assignment of a hyperplane to a node uniquely determines the periodization of the initial data to the node. In all cases, $S$-dimensional data is placed in the node for local computation.

- An $S$-dimensional DFT computation stage on the node level that computes the independent $R$-dimensional DFT of the initial data on the $S$-dimensional hyperplane assigned to the node.

- An uploading stage that places the computed data in its appropriate positions. Redundant data elimination must also be carried out.
8.4.1 An RTA for the 2-D Prime Case

In this section an implementation of the two-dimensional $p \times p$ RTA on a broadcast mode multiprocessor machine is given.

Recall the two-stage procedure of the RTA for a prime $p$ from Section 7.2.1.

**Reduction stage.** Compute the $p + 1$ summations:

$$g_j(k) = \sum_{l=0}^{p-1} f(k - lj, l), \quad j = 0, 1, \ldots, p - 1,$$

$$g_p(k) = \sum_{l=0}^{p-1} f(l, k).$$

**DFT stage.** Compute the $p + 1$ one-dimensional $p$-point DFTs:

$$F_\psi(l, j) = \sum_{k=0}^{p-1} g_j(k)e^{2\pi ikl/p}, \quad j = 0, 1, \ldots, p - 1$$

$$F_\psi(0, l) = \sum_{k=0}^{p-1} g_p(k)e^{2\pi ikl/p}.$$

The computation of the summation stage is realized by the following procedure (Rofheart [20]):

1. Assign the computation of $g_j, \ j = 0, 1, \ldots, M$ evenly among $M$ or fewer PEs, where $M = p + 1$ is the number of lines covering the $/p \times /p$ and initialize them with 0.

2. Broadcast the rows of the 2-D array $f$ of input data to the PEs. The PE assigned $g_p$ sums the elements of the $i$th row received and places that sum in $g_p(i)$. For each $g_j, \ j = 0, 1, \ldots, p - 1$ assigned to a PE, the $i$th row received is rotated $ji$ positions right and summed componentwise to the other received rows. Proceeding in this fashion, $g_j(d)$ will exist at location $d$ of the first row received. The rotation is achieved by an address offset and modulo $p$ address arithmetic; it requires no data movement.

3. In parallel, each PE computes a one-dimensional $p$-point FFT. These should be performed on the $p$ points $g_j(d), \ d = 0, 1, \ldots, p-1$ at the PE assigned the computation of $g_j$. After this stage the 2-D $p \times p$ DFT has been computed.

4. Upload the data. There is some redundancy in the data, which has to be removed and the data are permuted among the PEs. The redundancy
in this case is trivial, since \( f(0,0) \) is the only output common to every processor. The permutation is that every PE contains a line of output data as specified by Theorem 6.1.1. The row 0 and the column 0 are in the PEs which computed the \( g_0 \) and \( g_p \) terms, and are not permuted. Element \( l \) of the vector produced by the PE that computed \( g_j \) is the column \( l \) and the row \( lj \mod p \) element of the 2-D DFT.

The following algorithm is a pseudo code for the parallel computation of the RT algorithm. Assume that \( M \) PEs are available and assign the computation of \( g_j \) to PE \( j \).

**Algorithm 8.4.1 (RT Algorithm)**

\[
\text{\{Summation Stage\}} \\
\text{\{for } i = 0 \text{ to } p - 1 \text{ do} \\
\quad \text{broadcast row } i \text{ to all PEs;} \\
\quad \text{if } i \neq 0 \text{ then} \\
\quad \quad \text{for } j = 0 \text{ to } M \text{ do in parallel} \\
\quad \quad \quad \text{compute partial sum;}
\]

\[
\text{\{1-D } N \text{-point DFT Stage\}} \\
\text{\{for } j = 0 \text{ to } M - 1 \text{ do in parallel} \\
\quad \text{perform 1-D FFT } v_j = \text{FFT}(p)(g_j)
\]

\[
\text{\{Data Upload Stage\}} \\
\text{\{for } i = 0 \text{ to } M \text{ do} \\
\quad \text{if } i = P \text{ then \{column 0\}} \\
\quad \quad \text{for } t = 0 \text{ to } p - 1 \text{ do} \\
\quad \quad \quad F[0,t] = \text{report}(PE_i,v_t); \\
\quad \text{if } i = 0 \text{ then \{row 0\}} \\
\quad \quad \text{for } t = 1 \text{ to } p - 1 \text{ do} \\
\quad \quad \quad F[t,0] = \text{report}(PE_i,v_t); \\
\quad \text{if } 1 < i < P \text{ then} \\
\quad \quad \text{for } t = 1 \text{ to } p - 1 \text{ do} \\
\quad \quad \quad F[l,t] = \text{report}(PE_j,v_t);
\]

To obtain a measure of the performance of the algorithm, allow a comparison of a multiprocessor with \( p + 1 \) PEs running the parallelized \( p \times p \) RTA to a single processor computing the row column algorithm. Let the computational burden of the reduction stage be taken simultaneously with the data download. Each reduction operation for the 2-D \( p \times p \) case requires \( O(p) \) additions. Then the relative performance of the parallel line algorithm on \( p + 1 \) processors w. r. t. to
the row-column algorithm on a single processor is
\[
\text{speed-up} = \frac{2p \text{FFT}(p)}{\text{FFT}(p)} = 2p.
\]

The algorithm achieves a $2p$ speedup with $p + 1$ processors rather than the expected $p$ speedup because the number of 1-D DFT has been reduced from $2p$ to $p + 1$. The performance improvement of the algorithm depends entirely on the computation of the reduction operations in parallel with the data download.

Rofheert [20] also discusses parallel implementations for the 2-D power of prime case and the 3-D prime case.

### 8.5 Hybrid Algorithms

The RTA offers a trade-off between its degree of parallelism and its granularity depending on the dimension of output hyperplanes. The trade-off is limited by the transform size, which reduces its potential for scaling with machine parameters. Hybrid algorithms that nest the RTA within the multidimensional FFT will be discussed. The advantage of these hybrid algorithms is that both the dimension of the output hyperplanes and the size of the transform can be manipulated to affect a finer trade-off between degree of parallelism and granularity as compared to the stand-alone RTA.

The key idea of hybrid algorithms is to take a multidimensional FFT (either of the multidimensional Cooley-Tukey, vector-radix or Good-Thomas FFT will do), that operates at the global level to decompose the initial DFT computation into smaller-size DFT computations without change in dimension. The RTA can then be applied to each of theses smaller size DFTs. This permits the MFFT to be computed with a reduced degree of parallelism and no interprocessor communication.

Two hybrid algorithms will be presented. The first one uses the multidimensional Cooley-Tukey FFT for the global reduction stage and was developed by Rofheart [20]. The second one uses the multidimensional Good-Thomas FFT for the global reduction stage and was developed by Kechriotis et al. [15]. The second algorithm is not as general as the first one, but the authors present timing results on the Intel iPSC/860 Hypercube for the case $A = \left\{ N \times (M \times K \times P) \right\}$, for natural numbers $N, M, K$ and a prime number $P$, such that $\gcd(M, P) = \gcd(K, P) = 1$.

#### 8.5.1 The Cooley-Tukey/RTA Hybrid Algorithm

Rofheart [20] used the multidimensional Cooley-Tukey FFT for the global reduction stage to develop a hybrid algorithm. This algorithm will be presented below.
In Kronecker product formulation the multidimensional Cooley-Tukey factorization has the form

\[ F(N) = Q(F(M) \otimes I_L) T(I_M \otimes F(L)) P, \]  

(8.1)

where \( P \) and \( Q \) are permutation matrices, \( T \) is a diagonal matrix and

\[ N = (N_1, N_2, \ldots, N_R), \quad N = N_1N_2 \cdots N_R, \quad N_r = M_rL_r, \]
\[ M = (M_1, M_2, \ldots, M_R), \quad M = M_1M_2 \cdots M_R, \]
\[ L = (L_1, L_2, \ldots, L_R), \quad L = L_1L_2 \cdots L_R. \]

Stage 2 can be rewritten using the commutation property of stride permutations (Theorem 2.2.1) as

\[ F(M) \otimes I_L = P(N, M)(I_L \otimes F(M))P(N, L). \]  

(8.2)

Evaluation of the MDFT by the factorization of equation (8.1) requires computing stages composed of \( F(L) \) and \( F(M) \). It can be shown (see Rofheirt [20]) that the inputs to the \( i \)th \( F(M) \) of stage 2 are the collection of \( i \)th outputs of every \( F(L) \) of stage 1. Let that \( F(M) \) of stage 2 be assigned to a PE in a multiprocessor environment. The PE can evaluate that \( F(M) \) if the \( i \)th point from every \( F(L) \) of stage 1 is available at the PE. Below, an assignment of the RTA to the DFTs of stage 1 is given that satisfies this criterion.

Choose a dimension \( S \) such that the number of \( S \)-dimensional hyperplanes covering the \( R \)-dimensional index set for \( F(L) \) equals the degree of parallelism of the target machine. Assign one hyperplane to each node. Apply the RTA to a single \( F(L) \) computation. The result is that each node contains an \( S \)-dimensional hyperplane of the output of \( F(L) \). Continue this way to implement the remaining computations of \( F(L) \).

Each node contains the outputs of all the \( F(L) \) computations on the fixed pre-assigned \( S \)-dimensional hyperplane. The contents of a single node provide (after twiddle factor multiplications) the input into the second stage of the computation, which is completed by a single \( R \)-dimensional \( F(M) \) computation.

Different degrees of parallelism result for various factorizations of \( F(N) \) into \( F(L) \) and \( F(M) \) and the dimensionality \( S \) of the covering hyperplanes. These parameters can be matched to the degree of parallelism of the target machine.

Below, the parallel hybrid algorithm is presented in more detail with the help of an example, which is adopted from Rofheirt [20].

**Example 8.5.1 ( /9 \times /9 Cooley-Tukey/RTA Hybrid Algorithm)** Assume a 4 processor machine and consider the computation of a 9 \times 9 DFT. A direct mapping of the RTA
would require 12 processors (the number of lines covering \( A = 9 \times 9 \), see Theorem 6.1.5). However, the hybrid algorithm using a \( 3 \times 3 \) factorization requires only 4 processors. The factorization is given by

\[
F(N) = Q F(N, M)(I_L \otimes F(M))P(N, L) T(I_M \otimes F(L)) P,
\]

(8.3)

where \( P \) and \( Q \) are permutation matrices, \( T \) is a diagonal matrix and \( N = (9, 9), M = (3, 3), L = (3, 3) \).

The computation of Stage 1 depends on the evaluation of the \( 3 \times 3 \) DFTs \( F(3, 3) \) by the RTA. That computation is

\[
z = F(3, 3)x
\]

and is described in Section 7.2.1. The output array of the \( 3 \times 3 \) DFT is given by the four lines

\[
L(0, 1) = \{z_{0,0}, z_{0,1}, z_{0,2}\},
L(1, 1) = \{z_{0,0}, z_{1,1}, z_{2,2}\},
L(2, 1) = \{z_{0,0}, z_{2,1}, z_{2,1}\},
L(1, 0) = \{z_{0,0}, z_{1,0}, z_{2,0}\}.
\]

Each of these lines is assigned to a different processor. Let the output of the lines \( L(j, 1), j = 0, 1, 2 \) be computed by PE\(_j\), and the output on the line \( L(1, 0) \) computed by PE\(_3\). As the inputs are broadcast to the machine, each PE computes the reduction operation \( y_j \) associated with its assigned line of the output. After the broadcast is complete, each PE performs a 3-point DFT to produce its set of output points. Each of the nine \( 3 \times 3 \) DFTs of the first stage for the factorization (8.3) are computed in this way. The inputs to these DFTs are listed in Fig. 8.1. Each of the arrays is labeled by the index of its first element.

![Figure 8.1: Partitioning of the input elements for the 2-D \( /9 \times /9 \) RTA.](image)

The partition of the input elements results from decimating both dimensions of the array simultaneously. Each array of the partition is a coset of \( /9 \times /9 \) with respect to the subgroup \( 3 \times 3 /9 \). The label above each array is its coset leader. Throughout this example, the arrays will be defined and distinguished by their coset leaders.

Each of the nine required \( 3 \times 3 \) DFT of stage one are computed by the multiprocessor RTA as described above. The lines are taken over each of the cosets. The assignment of lines to PE's
is the same for each of the cosets. The PE assigned line \( L(0, 1) \) in coset \((0, 0)\) is assigned line \( L(0, 1) \) in every coset. For simplicity of presentation, allow the input data to be broadcast to the PEs by coset. Then the reduction operations are performed simultaneously with the data download. Each PE performs one addition for each word it inputs. As all the points of a coset are received, that associated reduction operation is completed. When the data is finished the PE performs nine independent 3-point 1-D DFT to complete stage one.

Consider the computation of the line \( L(0, 1) \) in each coset of the input partition. For each coset downloaded the PE computes the required reduction operation. When the download is complete, the PE performs the 1-D DFT. At this time the PE contains all the stage one outputs corresponding to the first row of each coset shown in Fig. 8.1 (enclosed in rectangles). The appropriate twiddle multiplications are applied at this time.

The second stage requires computing nine \( 3 \times 3 \) DFTs. To obtain all output points by the method described in this section, each PE must compute three \( 3 \times 3 \) DFTs. For the PE assigned line \( L(0, 1) \) of each stage one coset, one 2-D DFT is computed for each element in the line. That is, the first 2-D DFT is performed on all points in standard font \((x)\), the second is computed on all points in boldface \((x)\), and the third one is computed on all points in serif \((x)\) (see Fig. 8.1). After this stage the 2-D \( 9 \times 9 \) DFT is complete, the entire computation is distributed by output coset throughout the machine. This permutation is similar to the permutation that results from a typical application of the vector radix algorithm.

The factorization (8.1) of the \( 9 \times 9 \) DFT contains 18 \( 3 \times 3 \) DFTs. If each of these were evaluated by the row-column algorithm there would be 108 FFTs of 3-point each. In the hybrid algorithm every PE has to perform 9 3-point DFTs in Stage one and 3 \((3 \times 3)\)-point DFTs in stage two, giving a total of 27 3-point DFTs. Assume the reduction operations of the RTA in stage one are performed concurrently with the input data download and the communication time is not considered, then the computation speedup of the parallel hybrid algorithm relative to a single processor algorithm is

\[
\text{speed-up} = \frac{108 \text{ FFT}(3)}{27 \text{ FFT}(3)} = 4,
\]

which is the ideal linear speedup, and requires no interprocessor communication.

### 8.5.2 The Good-Thomas/RTA Hybrid Algorithm

In Keckriotis et al. [15] a hybrid algorithm of the Good-Thomas prime factor FFT and the RTA is introduced. Implementation issues on an Intel iPSC/860 hypercube are discussed and timing results for large 2-D and 3-D DFTs with index sets \( /MP \times /KP \) and \( /N \times /MP \times /KP \), respectively are provided, where \( N, M, K \) are powers of two and \( P \) is a small prime number such as 3, 5, or 7.

The derivation of the algorithm is easy, when using Kronecker product notation. Let \( A = /N \times /MP \times /KP \) for natural numbers \( N, M, K \) and a prime number \( P \), such that \( \gcd(M, P) = \gcd(K, P) = 1 \). In Kronecker product formulation the Fourier transform on \( A \) has the form

\[
F(N, MP, KP) = F(N) \otimes F(MP) \otimes F(KP).
\] (8.4)
By applying the Good-Thomas FFT to the factors $F(MP)$ and $F(KP)$, equation (8.4) equals
\[ F(N)(Q'(F(M) \otimes F(P))Q) \otimes (Q'(F(K) \otimes F(P))Q), \]  
where $Q$ and $Q'$ resp. $\bar{Q}$ and $\bar{Q}'$ are permutation matrices according to the Good-Thomas factorization of $MP$ resp. $KP$ given in Theorem 5.1.2. By applying the multiplicative property (Property 2.1.4), equation (8.5) equals
\[ (I_N \otimes Q' \otimes \bar{Q}')(F(N) \otimes F(M) \otimes F(P) \otimes F(K) \otimes F(P))\)(I_N \otimes Q \otimes \bar{Q}). \]  
By applying the permutation theorem (Theorem 2.2.2) equation (8.6) becomes
\[ R'(F(N) \otimes F(M) \otimes F(K) \otimes F(P) \otimes F(P))R, \]  
where
\[ R' = (I_N \otimes Q' \otimes \bar{Q}')P^{-1}(N, M, P, K, P), \]
\[ R = (I_N \otimes Q \otimes \bar{Q})P_{\sigma}(N, M, P, K, P), \]
and $\sigma$ is the permutation $(1, 2, 4, 3, 5)$.

The expression in parenthesis is the Fourier transform on $A = . /N \times . /M \times . /K \times . /P \times . /P$. Now the RTA derived in Section 7.2.3 for $k = 4$ and $N = 5$ and its parallel implementation discussed is Section 8.4 can be applied. It is interesting to note that the reindexing defined by the matrices $R$ and $R'$ can now be incorporated in the input periodizations and be performed during data upload.

**Numerical Results**

Kechriotis et al. [15] have implemented the hybrid RTA (HRTA) described above on an Intel iPSC/860 hypercube and compared it to the highly optimized Intel iPSC/860 vendor supplied RC implementation for large 2-D and 3-D DFTs with index sets . /MP×. /KP and . /N×. /MP×. /KP respectively, where $N, M, K$ are powers-of-two and $P = 3$ using $P + 1 = 4$ nodes. These results, together with a short description of the Intel iPSC/860 hypercube, are presented below.

The Intel iPSC/860 hypercube parallel processing system was a distributed memory, multiple instruction multiple data (MIMD) hypercube, containing up to 128 = 2$^7$ processing elements (PEs) based on the Intel i860 high performance 64-bit RISC microprocessor. The i860 had a peak performance of 80 Mflop/s and was equipped with 8K data and 4K instruction cache memory. Each node had 8 to 64 Mbytes of external local memory, a network interface and a message router. The router handled up to eight bidirectional communication channels. Seven of them could be connected to neighboring nodes and one of them was dedicated to external I/O and was directly connected to the host processor.
Execution times of the RC and the HRTA were measured from the point that all the necessary data already reside in the nodes, and until the results have been computed and stored in the processors' local memory, since data input and output depends on the specific application. In both implementations the distribution of the results was different from the original distribution. Using the HRTA the results were distributed along hyperplanes assigned to each processor and using the RC method the results were distributed in a transposed fashion.

<table>
<thead>
<tr>
<th>Hybrid RTA</th>
<th>RC Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MP x KP</strong></td>
<td><strong>time</strong></td>
</tr>
<tr>
<td>192 x 192</td>
<td>67 ms</td>
</tr>
<tr>
<td>192 x 384</td>
<td>131 ms</td>
</tr>
<tr>
<td>384 x 384</td>
<td>255 ms</td>
</tr>
<tr>
<td>384 x 768</td>
<td>512 ms</td>
</tr>
<tr>
<td>768 x 768</td>
<td>1117 ms</td>
</tr>
</tbody>
</table>

**Table 8.1:** Performance of the 2-D hybrid RTA (HRTA) parallel algorithm vs. the iPSC/860 optimized RC parallel code.

<table>
<thead>
<tr>
<th>Hybrid RTA</th>
<th>RC Method</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>MP x KP</strong></td>
<td><strong>time</strong></td>
</tr>
<tr>
<td>8 x 96 x 96</td>
<td>184 ms</td>
</tr>
<tr>
<td>8 x 96 x 192</td>
<td>352 ms</td>
</tr>
<tr>
<td>8 x 192 x 192</td>
<td>719 ms</td>
</tr>
<tr>
<td>8 x 192 x 384</td>
<td>1522 ms</td>
</tr>
<tr>
<td>16 x 96 x 96</td>
<td>338 ms</td>
</tr>
<tr>
<td>16 x 96 x 192</td>
<td>690 ms</td>
</tr>
<tr>
<td>16 x 192 x 192</td>
<td>1422 ms</td>
</tr>
<tr>
<td>4 x 384 x 384</td>
<td>1719 ms</td>
</tr>
</tbody>
</table>

**Table 8.2:** Performance of the 3-D hybrid RTA (HRTA) parallel algorithm vs. the iPSC/860 optimized RC parallel code.

In Tables 8.1 and 8.2 the runtimes of the two algorithms for a variety of data sizes are presented. It can be seen, that the RC is up to 70\% slower than the HRTA, depending on the amount of zero padding. Table 8.2 also shows the computational times required for the DFT of the same data set using the RC on 8 nodes. It is interesting to observe that even if the number of nodes is doubled the performance improvement amounts only to an average of 17\% relative to the four-node HRTA implementation.

The numerical results show that the RTA and especially the HRTA perform very well on parallel environments since they require no interprocessor communication.
However, the results presented in Kechriotis et al. [15] have two drawbacks. First, the implementation of the HRTA is not able to compute the periodizations during the data download, i.e., no overlapping of communication and computation is done. Second, the Intel vendor supplied FFT routines only work for transform sizes which are powers of two. That means in the test cases zero padding is required.
Chapter 9

Conclusion

In this work, an abstract approach to define the discrete Fourier (DFT) transform has been taken. A finite Abelian group serves as the indexing set for the DFT. The advantage of this approach is that it removes the dependence on the coordinates and solely references the additive structure of the Abelian group. This is especially useful in a multidimensional setting.

It was noted in Chapter 8 that on many computers, the dominating factor in the performance of the FFT is the memory access pattern—the dataflow—of the algorithm. Therefore it is necessary to develop algorithms that have an optimal dataflow with respect to the machine’s memory hierarchy. This is especially true in parallel environments having a local and a global memory.

Two different ways to address the problem of dataflow have been introduced. Both of them use the theory of Fourier transforms on finite Abelian groups.

The first one is the so called dimensionless FFT (Section 5.3), which is able to compute all multidimensional DFTs with the same number of input points. Changing the dimension is achieved by relabeling the input and the output and changing the twiddle factors. The dimensionless theorem has two important consequences. First, it permits the design of a device with a hardwired dataflow that is able to compute the DFT for a fixed number of input points and for any appropriate dimension. Second, it suggests that an effective means of implementing and optimizing FFT algorithms is to search for an optimal dataflow and then instantiate the dataflow, by providing appropriate twiddle factors and data relabelings.

The second way to address the problem of dataflow are reduced transform algorithms (RTAs), which were introduced in Chapter 7. These algorithms reduce the computation of the DFT on a finite Abelian group $A$ to independent computations of DFTs on certain subgroups of $A$. In the two-dimensional $p \times p$ case, the number of one-dimensional DFTs that are needed to perform the two-dimensional DFT can be reduced to $p + 1$ compared to $2p$ in the row-column algorithm. Nevertheless, the overall complexity of RTAs is higher because in the reduction stage $O(p^3)$ additions have to be performed.

RTAs are especially useful in parallel environments, since they require no interprocessor communication and the preadditions can be incorporated into the uploading of the data to the processing elements (overlapping of communication and computation). The scalability of RTAs running on parallel environments is
limited by the number of hyperplanes covering the index set. Therefore hybrid algorithms that nest the RTA within a multidimensional FFT have been introduced in Section 8.5. They allow a finer trade-off between degree of parallelism and granularity and still do not require interprocessor communication.

**Future Work**

It has been demonstrated, that the abstract approach to discrete Fourier transforms provides powerful tools for developing new algorithms. Reduced transform algorithms for multidimensional DFTs are very flexible, because their degree of parallelism can be matched to the target architecture in parallel environments.

An idea for future work is to incorporate this algorithm into coming versions of FFTW, the “Fastest Fourier Transform in the West” (Frigo and Johnson [6]). FFTW is a C subroutine library for computing the DFT in one or more dimensions, of both real and complex data, and of arbitrary input size. The package is highly portable and architecture adaptive, in the sense, that it is able to choose the best performing kernels for a given architecture based on run-time tests. Parallelized versions of FFTW are also available.

The upcoming version will allow the user to use his own algorithms for computing the DFT. The package will then test the algorithm’s runtime on the specific machine and will use it, if it is faster than the standard algorithms. Sophisticated implementations of RTAs will most likely outperform standard algorithms for certain combinations of transform sizes and computer architecture.
Bibliography


http://citeseer.nj.nec.com/106774.html


