国际著名数学图书——影印版

Computational Frameworks for the Fast Fourier Transform

快速傅里叶变换的计算框架

Charles Van Loan 著

TSINGHUA UNIVERSITY PRESS

siam

Computational Frameworks for the Fast Fourier Transform

快速傅里叶变换的计算框架

Charles Van Loan　著

TSINGHUA UNIVERSITY PRESS
北京

siam

Charles Van Loan
Computational Frameworks for the Fast Fourier Transform
ISBN:0-898712-85-8

图书在版编目（CIP）数据

快速傅里叶变换的计算框架 = Computational Frameworks for the Fast Fourier Transform：英文/（美）罗恩（Loan C. V.）著.-- 影印本. -- 北京：清华大学出版社，2011.2
（国际著名数学图书）
ISBN 978-7-302-24497-4

Ⅰ. ①快…　Ⅱ. ①罗…　Ⅲ. ①傅里叶变换–英文　Ⅳ. ①O174.22

中国版本图书馆CIP数据核字（2011）第006586号

责任编辑：陈朝晖
责任印制：王秀菊
出版发行：清华大学出版社　　**地　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　编**：100084
社　总　机：010-62770175　　**邮　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质　量　反　馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印　装　者：北京国马印刷厂
经　销：全国新华书店
开　本：175×245　**印　张**：18.25
版　次：2011 年 2 月第 1 版　　**印　次**：2011 年 2 月第1 次印刷
印　数：1～3000
定　价：39.00 元

产品编号：027808-01

Preface

The fast Fourier transform (FFT) is one of the truly great computational developments of this century. It has changed the face of science and engineering so much so that it is not an exaggeration to say that *life as we know it would be very different without the FFT.*

Unfortunately, the simplicity and intrinsic beauty of many FFT ideas are buried in research papers that are rampant with vectors of subscripts, multiple summations, and poorly specified recursions. The poor mathematical and algorithmic notation has retarded progress and has led to a literature of duplicated results. I am convinced that *life as we know it would be considerably different if, from the 1965 Cooley–Tukey paper onwards, the FFT community had made systematic and heavy use of matrix-vector notation!* Indeed, by couching results and algorithms in matrix/vector notation, the FFT literature can be unified and made more understandable to the outsider. The central theme in this book is the idea that different FFT algorithms correspond to different factorizations of the discrete Fourier transform (DFT) matrix. The matrix factorization point of view, so successful in other areas of numerical linear algebra, goes a long way toward unifying and simplifying the FFT literature. It closes the gap between the computer implementation of an FFT and the underlying mathematics, because it forces us to think well above the scalar level.

By approaching the FFT matrix/vector terms, the algorithm can be used as a vehicle for studying key aspects of advanced scientific computing, e.g., vectorization, locality of reference, and parallelization. The FFT deserves to be ranked among the great teaching algorithms in computational science such as Gaussian elimination, the Lanczos process, binary search, etc.

I would like to thank a number of colleagues and students for helping to make this book possible. This manuscript began as a set of class notes for a vector-parallel FFT course taught at Cornell in the spring of 1987. The contributions of all the students, especially Clare Chu, Yi-Zhong Wu, Anne Elster, and Nihal Wijeyesekera, are gratefully acknowledged. Clare Chu went on to write a beautiful thesis in the area that has been of immense help to me during the writing of §§1.4, 3.5, 4.3, and 4.4.

Sid Burrus's March 1990 visit to Cornell rejuvenated my interest in the manuscript, which at that time had been dormant for a number of years. I arranged to give a second edition of the FFT course during the fall of that year. Among the attendees who put up with my revisions and expansions of the earlier manuscript, Richard Huff, Wei Li, Marc Parmet, and Stephan Parrett deserve special mention for their contributions to the finished volume. I am also indebted to Dave Bond, Bill Campbell, Greg Henry, Ricardo Pomeranz, Dave Potyondy, and Dean Robinson.

I also wish to thank Chris Paige and Clement Pellerin of McGill University and Lennart Johnsson of Thinking Machines for reading portions of the text and making many corrections and valuable suggestions.

In addition, I would like to acknowledge the financial suppport of the Army Research Office through the Mathematical Sciences Institute (MSI) at Cornell. With the help of the MSI, I was able to organize a workshop on the FFT during the spring of 1987. More recently, I was supported during the summer of 1990 by grants to the Cornell Computational Optimization Project from the Office of Naval Research and the National Science Foundation. During this time, the manuscript was considerably refined.

Finally, I am indebted to Cindy Robinson–Hubbell at the Advanced Computing Research Institute at Cornell for overseeing the production of the camera-ready copy and to Crystal Norris at SIAM for a superb job of copyediting and managing the whole project.

Preliminary Remarks

References

Annotated bibliographies are given in each section and a master list of references is supplied at the end of the book. We offer a few general bibliographic remarks here at the outset, beginning with the following list of good background texts:

R.E. Blahut (1984). *Fast Algorithms for Digital Signal Processing,* Addison-Wesley, Reading, MA.

R.N. Bracewell (1978). *The Fourier Transform and Its Applications,* McGraw-Hill, New York.

E.O. Brigham (1974). *The Fast Fourier Transform,* Prentice-Hall, Englewood Cliffs, NJ.

E.O. Brigham (1988). *The Fast Fourier Transform and Its Applications,* Prentice-Hall, Englewood Cliffs, NJ.

C.S. Burrus and T. Parks (1985). *DFT/FFT and Convolution Algorithms,* John Wiley & Sons, New York.

H.J. Nussbaumer (1981b). *Fast Fourier Transform and Convolution Algorithms,* Springer-Verlag, New York.

The bibliography in Brigham (1988) is particularly extensive. The researcher may also wish to browse through the 2000+ entries in

M.T. Heideman and C.S. Burrus (1984). "A Bibliography of Fast Transform and Convolution Algorithms," Department of Electrical Engineering, Technical Report 8402, Rice University, Houston, TX.

A particularly instructive treatment of the Fourier transform may be found in

G. Strang (1987). *Introduction to Applied Mathematics,* Wellesley-Cambridge Press, Wellesley, MA.

The books by Brigham contain ample pointers to many engineering applications of the FFT. However, the technique also has a central role to play in many "traditional" areas of applied mathematics. See

J.R. Driscoll and D.M. Healy Jr. (1989). "Asymptotically Fast Algorithms for Spherical and Related Transforms," Technical Report PCS-TR89-141, Department of Mathematics and Computer Science, Dartmouth College.

M.H. Gutknecht (1979). "Fast Algorithms for the Conjugate Periodic Function," *Computing 22,* 79–91.

P. Henrici (1979). "Fast Fourier Methods in Computational Complex Analysis," *SIAM Rev. 21,* 460–480.

T.W. Körner (1988). *Fourier Analysis,* Cambridge University Press, New York.

The FFT has an extremely interesting history. For events surrounding the publication of the famous 1965 paper by Cooley and Tukey, see

J.W. Cooley (1987). "How the FFT Gained Acceptance," in *History of Scientific Computing*, S. Nash (ed.), ACM Press, Addison-Wesley, Reading, MA.

J.W. Cooley, R.L. Garwin, C.M. Rader, B.P. Bogert, and T.G. Stockham Jr. (1969). "The 1968 Arden House Workshop on Fast Fourier Transform Processing," *IEEE Trans. Audio Electroacoustics AU-17*, 66–76.

J.W. Cooley, P.A. Lewis, and P.D. Welch (1967). "Historical Notes on the Fast Fourier Transform," *IEEE Trans. Audio and Electroacoustics AU-15*, 76–79.

However, the true history of the FFT idea begins with (who else!) Gauss. For a fascinating account of these developments read

M.T. Heideman, D.H. Johnson, and C.S. Burrus (1985). "Gauss and the History of the Fast Fourier Transform," *Arch. Hist. Exact Sci. 34*, 265–277.

History of Factorization Ideas

The idea of connecting factorizations of the DFT matrix to FFT algorithms has a long history and is *not* an idea novel to the author. These connections are central to the book and deserve a chronologically ordered mention here at the beginning of the text:

I. Good (1958). "The Interaction Algorithm and Practical Fourier Analysis," *J. Roy. Stat. Soc. Ser. B, 20*, 361–372. Addendum, *J. Roy. Stat. Soc. Ser. B, 22*, 372–375.

E.O. Brigham and R.E. Morrow (1967). "The Fast Fourier Transform," *IEEE Spectrum 4*, 63–70.

W.M. Gentleman (1968). "Matrix Multiplication and Fast Fourier Transforms," *Bell System Tech. J. 47*, 1099–1103.

F. Theilheimer (1969). "A Matrix Version of the Fast Fourier Transform," *IEEE Trans. Audio and Electroacoustics AU-17*, 158–161.

D.K. Kahaner (1970). "Matrix Description of the Fast Fourier Transform," *IEEE Trans. Audio and Electroacoustics AU-18*, 442–450.

M.J. Corinthios (1971). "The Design of a Class of Fast Fourier Transform Computers," *IEEE Trans. Comput. C-20*, 617–623.

M. Drubin (1971a). "Kronecker Product Factorization of the FFT Matrix," *IEEE Trans. Comput. C-20*, 590–593.

I.J. Good (1971). "The Relationship between Two Fast Fourier Transforms," *IEEE Trans. Comput. C-20*, 310–317.

P.J. Nicholson (1971). "Algebraic Theory of Finite Fourier Transforms," *J. Comput. System Sci. 5*, 524–527.

A. Rieu (1971). "Matrix Formulation of the Cooley and Tukey Algorithm and Its Extension," *Revue Cethedec 8*, 25–35.

B. Ursin (1972). "Matrix Formulations of the Fast Fourier Transform," *IEEE Comput. Soc. Repository R72-42*.

H. Sloate (1974). "Matrix Representations for Sorting and the Fast Fourier Transform," *IEEE Trans. Circuits and Systems CAS-21*, 109–116.

D.J. Rose (1980). "Matrix Identities of the Fast Fourier Transform," *Linear Algebra Appl. 29*, 423–443.

C. Temperton (1983). "Self-Sorting Mixed Radix Fast Fourier Transforms," *J. Comput. Phys. 52*, 1–23.

V.A. Vlasenko (1986). "A Matrix Approach to the Construction of Fast Multidimensional Discrete Fourier Transform Algorithms," *Radioelectron. and Commun. Syst. 29*, 87–90.

D. Rodriguez (1987). *On Tensor Product Formulations of Additive Fast Fourier Transform Algorithms and Their Implementations*, Ph.D. Thesis, Department of Electrical Engineering, The City College of New York, CUNY.

R. Tolimieri, M. An, and C. Lu (1989). *Algorithms for Discrete Fourier Transform and Convolution*, Springer-Verlag, New York.

H.V. Sorensen, C.A. Katz, and C.S. Burrus (1990). "Efficient FFT Algorithms for DSP Processors Using Tensor Product Decomposition," *Proc. ICASSP-90*, Albuquerque, NM.

J. Johnson, R.W. Johnson, D. Rodriguez, and R. Tolimieri (1990). "A Methodology for Designing, Modifying, and Implementing Fourier Transform Algorithms on Various Architectures," *Circuits, Systems, and Signal Processing 9*, 449–500.

Software

The algorithms in this book are formulated using a stylized version of the Matlab language which is very expressive when it comes to block matrix manipulation. The reader may wish to consult

M. Metcalf and J. Reid (1990). *Fortran 90 Explained*, Oxford University Press, New York

for a discussion of some very handy array capabilities that are now part of the modern Fortran language and which fit in nicely with our algorithmic style.

We stress that nothing between these covers should be construed as even approximate production code. The best reference in this regard is to the package *FFTPACK* due to Paul Swarztrauber and its vectorized counterpart, *VFFTPACK* due to Roland Sweet. This software is available through netlib. A message to netlib@ornl.gov with a message of the form "send index for FFTPACK" or "send index for VFFTPACK" will get you started.

Contents

Chapter 1

The Radix-2 Frameworks

A fast Fourier transform (FFT) is a quick method for forming the matrix-vector product $F_n x$, where F_n is the discrete Fourier transform (DFT) matrix. Our examination of this area begins in the simplest setting: the case when $n = 2^t$. This permits the orderly repetition of the central divide-and-conquer process that underlies all FFT work. Our approach is based upon the factorization of F_n into the product of $t = \log_2 n$ sparse matrix factors. Different factorizations correspond to different *FFT frameworks*. Within each framework different *implementations* are possible.

To navigate this hierarchy of ideas, we rely heavily upon block matrix notation, which we detail in §1.1. This "language" revolves around the Kronecker product and is used in §1.2 to establish the "radix-2 splitting," a factorization that indicates how a $2m$-point DFT can be rapidly determined from a pair of m-point DFTs. The repeated application of this process leads to our first complete FFT procedure, the famous Cooley–Tukey algorithm in §1.3.

Before fully detailing the Cooley–Tukey process, we devote §§1.4 and 1.5 to a number of important calculations that surface in FFT work. These include the butterfly computation, the generation of weights, and various data transpositions. Using these developments, we proceed to detail the *in-place* Cooley–Tukey framework in (§1.6). In-place FFT procedures overwrite the input vector with its DFT without making use of an additional vector workspace. However, certain data permutations are involved that are sometimes awkward to compute. These permutations can be avoided at the expense of a vector workspace. The resulting *in-order* FFTs are discussed in §1.7, where a pair of *autosort* frameworks are presented. A fourth framework due to Pease is covered in §1.8.

Associated with each of these four methods is a different factorization of the DFT matrix. As we show in §1.9, four additional frameworks are obtained by transposing these four factorizations. A fast procedure for the inverse DFT can be derived by conjugating any of the eight "forward" DFT factorizations.

1.1 Matrix Notation and Algorithms

The DFT is a matrix-vector product that involves a highly structured matrix. A description of this structure requires the language of block matrices with an emphasis on Kronecker products. The numerous results and notational conventions of this section are used throughout the text, so it is crucial to have a facility with what follows.

1.1.1 Matrix/Vector Notation

We denote the vector space of complex n-vectors by $\mathbb{C}^n$, with components indexed from zero unless otherwise stated. Thus,

$$x \in \mathbb{C}^2 \quad \Rightarrow \quad x = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}.$$

For m-by-n complex matrices we use the notation $\mathbb{C}^{m \times n}$. Rows and columns are indexed from zero, e.g.,

$$A \in \mathbb{C}^{2\times 3} \quad \Rightarrow \quad A = \begin{bmatrix} a_{00} & a_{01} & a_{02} \\ a_{10} & a_{11} & a_{12} \end{bmatrix}.$$

From time to time exceptions to the index-from-zero rule are made. These occasions are rare and the reader will be amply warned prior to their occurrence.

Real n-vectors and real m-by-n matrices are denoted by $\mathbb{R}^n$ and $\mathbb{R}^{m\times n}$, respectively.

If $A = (a_{kj})$, then we say that a_{kj} is the (k, j) entry of A. Sometimes we say the same thing with the notation $[A]_{kj}$. Thus, if α is a scalar then $[\alpha A]_{kj} = \alpha a_{kj}$.

The *conjugate*, the *transpose*, and the *conjugate transpose* of an m-by-n complex matrix $A = (a_{kj})$ are denoted as follows:

$$\begin{array}{lcll} \bar{A} & = & (\bar{a}_{kj}) & (m\text{-by-}n), \\ A^T & = & (a_{jk}) & (n\text{-by-}m), \\ A^H & = & (\bar{a}_{jk}) & (n\text{-by-}m). \end{array}$$

The *inverse* of a nonsingular matrix $A \in \mathbb{C}^{n\times n}$ is designated by A^{-1}. The n-by-n identity is denoted by I_n. Thus, $AA^{-1} = A^{-1}A = I_n$.

1.1.2 The Discrete Fourier Transform

The *discrete Fourier transform* (DFT) on $\mathbb{C}^n$ is a matrix-vector product. In particular, $y = [y_0, \ldots, y_{n-1}]^T$ is the DFT of $x = [x_0, \ldots, x_{n-1}]^T$ if for $k = 0, \ldots, n-1$ we have

$$y_k = \sum_{j=0}^{n-1} \omega_n^{kj} x_j, \tag{1.1.1}$$

where

$$\omega_n = \cos(2\pi/n) - i\sin(2\pi/n) = \exp(-2\pi i/n) ,$$

and $i^2 = -1$. Note that ω_n is an nth root of unity: $\omega_n^n = 1$.

In the FFT literature, some authors set $\omega_n = \exp(2\pi i/n)$. This does not seriously affect any algorithmic or mathematical development. Another harmless convention that we have adopted is *not* to scale the summation in (1.1.1) by $1/n$, a habit shared by other authors.

In matrix-vector terms, the DFT as defined by (1.1.1) is prescribed by

$$y = F_n x,$$

where

$$F_n = (f_{pq}) \qquad f_{pq} = \omega_n^{pq} = \exp(-2\pi pqi/n) \tag{1.1.2}$$

is the n-by-n *DFT matrix*. Thus,

$$F_1 = [\,1\,], \qquad F_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad \text{and} \quad F_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & -1 & i \\ 1 & -1 & 1 & -1 \\ 1 & i & -1 & -i \end{bmatrix}.$$

1.1.3 Some Properties of the DFT Matrix

We say that $A \in \mathbb{C}^{n\times n}$ is *symmetric* if $A^T = A$ and *Hermitian* if $A^H = A$. As an exercise in our notation, let us prove a few facts about F_n that pertain to these properties. The first result concerns some obvious symmetries.

Theorem 1.1.1 *The matrix F_n is symmetric.*

Proof. $[F_n^T]_{jk} = \omega_n^{kj} = \omega_n^{jk} = [F_n]_{jk}$. □

Note that if $n > 2$, then F_n is *not* Hermitian. Indeed, $F_n^H = \bar{F}_n^T = \bar{F}_n$. However, if $n = 1$ or 2, then F_n is real and so F_1 and F_2 are Hermitian.

Two vectors $x, y \in \mathbb{C}^n$ are *orthogonal* if $x^H y = 0$. The next result shows that the columns of F_n are orthogonal.

Theorem 1.1.2 $F_n^H F_n = nI_n$.

Proof. If μ_{pq} is the inner product of columns p and q of F_n, then by setting $\omega = \omega_n$ we have

$$\mu_{pq} = \sum_{k=0}^{n-1} (\bar{\omega}^{kp})\omega^{kq} = \sum_{k=0}^{n-1} \omega^{k(q-p)}.$$

If $q = p$, then $\mu_{pq} = n$. Otherwise, $\omega^{q-p} \neq 1$, and it follows from the equation

$$(1 - \omega^{q-p})\mu_{pq} = \sum_{k=0}^{n-1} \omega^{k(q-p)} - \sum_{k=1}^{n} \omega^{k(q-p)} = 1 - \omega^{n(q-p)} = 0$$

that $\mu_{pq} = 0$ whenever $q \neq p$. □

We say that $Q \in \mathbb{C}^{n\times n}$ is *unitary* if $Q^{-1} = Q^H$. From Theorem 1.1.2 we see that $F_n/\sqrt{n}$ is unitary. Thus, the DFT is a scaled unitary transformation on $\mathbb{C}^n$.

1.1.4 Column and Row Partitionings

If $A \in \mathbb{C}^{m\times n}$ and we designate the kth column by a_k, then

$$A = [\, a_0 \mid a_1 \mid \cdots \mid a_{n-1} \,], \qquad a_k \in \mathbb{C}^m,$$

is a *column partitioning* of A. Thus, if $A = F_n$ and $\omega = \omega_n$, then

$$a_k = \begin{bmatrix} 1 \\ \omega^k \\ \vdots \\ \omega^{k(n-1)} \end{bmatrix}.$$

Likewise,

$$A = \begin{bmatrix} b_0^T \\ b_1^T \\ \vdots \\ b_{m-1}^T \end{bmatrix}, \qquad b_k \in \mathbb{C}^n,$$

is a *row partitioning* of $A \in \mathbb{C}^{m\times n}$. Again, if $A = F_n$, then

$$b_k^T = \left[1,\, \omega^k, \ldots, \omega^{k(n-1)}\right].$$

1.1.5 Submatrix Specification

Submatrices of $A \in \mathbb{C}^{m\times n}$ are specified by the notation $A(u,v)$, where u and v are integer row vectors that "pick out" the rows and columns of A that define the submatrix. Thus, if $u = [\,0\ 2\,]$, $v = [\,0\ 1\ 3\,]$, and $B = A(u,v)$, then

$$B = \begin{bmatrix} a_{00} & a_{01} & a_{03} \\ a_{20} & a_{21} & a_{23} \end{bmatrix}.$$

Sufficiently regular index vectors u and v can be specified with the *colon notation*:

$$u = k{:}j \quad \Leftrightarrow \quad u = [\, k,\ k+1, \ldots, j] \qquad k \le j.$$

Thus, $A(2{:}4, 3{:}7)$ is a 3-by-5 submatrix defined by rows 2, 3, and 4 and columns 3, 4, 5, 6, and 7. There are some special conventions when entire rows or columns are to be extracted from the parent matrix. In particular, if $A \in \mathbb{C}^{m\times n}$, then

$$\begin{aligned} A(u,:) &\Leftrightarrow A(u, 0{:}n-1), \\ A(:,v) &\Leftrightarrow A(0{:}m-1, v). \end{aligned}$$

Index vectors with nonunit increments may be specified with the notation $i{:}j{:}k$, meaning count from i to k in steps of length j. This means that $A(0{:}2{:}m-1, :)$ is made up of A's even-indexed rows, whereas $A(:, n-1{:}-1{:}0)$ is A with its columns in reverse order. If $A = [\, a_0 \mid a_1 \mid \cdots \mid a_{21} \,]$, then $A(:, 3{:}4{:}21) = [\, a_3 \mid a_7 \mid a_{11} \mid a_{15} \mid a_{19} \,]$.

1.1.6 Block Matrices

Column or row partitionings are examples of matrix blockings. In general, when we write

$$A = \begin{array}{c} \left[\begin{array}{ccc} A_{00} & \dots & A_{0,q-1} \\ \vdots & & \vdots \\ A_{p-1,0} & \dots & A_{p-1,q-1} \end{array}\right] \\ \begin{array}{ccc} n_0 & \dots & n_{q-1} \end{array} \end{array} \begin{array}{c} m_0 \\ \vdots \\ m_{p-1} \\ \end{array} \qquad (1.1.3)$$

we are choosing to regard A as a p-by-q block matrix where the block A_{kj} is an m_k-by-n_j matrix of scalars.

The notation for block vectors is similar. Indeed, if $q = 1$ in (1.1.3), then we say that A is a p-by-1 block vector.

The manipulation of block matrices is analogous to the manipulation of scalar matrices. For example, if $A = (A_{ij})$ is a p-by-q block matrix and $B = (B_{ij})$ is a q-by-r block matrix, then $AB = C = (C_{kj})$ can be regarded as a p-by-r block matrix with

$$C_{kj} = A_{k0}B_{0j} + \cdots A_{k,q-1}B_{q-1,j},$$

assuming that all of the individual matrix-matrix multiplications are defined.

Partitioning a matrix into equally sized blocks is typical in FFT derivations. Thus, if $m_0 = \cdots = m_{p-1} = \nu$ and $n_0 = \cdots = n_{q-1} = \eta$ in equation (1.1.3), then $A_{kj} = A(k\nu{:}(k+1)\nu - 1, j\eta{:}(j+1)\eta - 1)$.

1.1.7 Regarding Vectors as Arrays

The reverse of regarding a matrix as an array of vectors is to take a vector and arrange it into a matrix. If $x \in \mathbb{C}^n$ and $n = rc$, then by $x_{r\times c}$ we mean the matrix

$$x_{r\times c} = [\, x(0{:}r-1) \mid x(r{:}2r-1) \mid \cdots \mid x(n-r{:}n-1) \,] \in \mathbb{C}^{r\times c},$$

i.e., $[x_{r\times c}]_{kj} = x_{jr+k}$. For example, if $x \in \mathbb{C}^{12}$, then

$$x_{3\times 4} = \begin{bmatrix} x_0 & x_3 & x_6 & x_9 \\ x_1 & x_4 & x_7 & x_{10} \\ x_2 & x_5 & x_8 & x_{11} \end{bmatrix}.$$

The colon notation is also handy for describing the columns of $x^T_{r\times c}$:

$$x^T_{r\times c} = [\, x(0{:}r{:}n-1) \mid x(1{:}r{:}n-1) \mid \cdots \mid x(r-1{:}r{:}n-1) \,].$$

Thus,

$$x^T_{3\times 4} = \begin{bmatrix} x_0 & x_1 & x_2 \\ x_3 & x_4 & x_5 \\ x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} \end{bmatrix} = [\, x(0{:}3{:}11) \mid x(1{:}3{:}11) \mid x(2{:}3{:}11) \,].$$

Suppose $x \in \mathbb{C}^n$ with $n = rc$. In the context of an algorithm, it may be useful for the sake of clarity to identify $x \in \mathbb{C}^n$ with a doubly subscripted array $X(0{:}r-1, 0{:}c-1)$. Consider the operation $b \leftarrow b + x_{r\times c}a$, where $a \in \mathbb{C}^c$, $b \in \mathbb{C}^r$, and "$\leftarrow$" designates assignment. The doubly subscripted array formulation

```
$X(0{:}r-1, 0{:}c-1) \equiv x_{r\times c}$
for $k = 0{:}r-1$
    for $j = 0{:}c-1$
        $b(k) \leftarrow b(k) + X(k,j)a(j)$
    end
end
```

is certainly clearer than the following algorithm, in which x is referenced as a vector:

```
for $k = 0{:}r-1$
    for $j = 0{:}c-1$
        $b(k) \leftarrow b(k) + x(jr+k)a(j)$
    end
end
```

Our guiding philosophy is to choose a notation that best displays the underlying computations.

Finally, we mention that a vector of length $n = n_1 \cdots n_t$ can be identified with a t-dimensional array $X(0{:}n_1-1, \ldots, 0{:}n_t-1)$ as follows:

$$X(\alpha_1, \ldots, \alpha_t) \equiv x(\alpha_1 + \alpha_2 N_2 + \cdots + \alpha_t N_t).$$

Here, $N_k = n_1 \cdots n_{k-1}$. In multidimensional settings such as this, components are indexed from one.

1.1.8 Permutation Matrices

An n-by-n permutation matrix is obtained by permuting the columns of the n-by-n identity I_n. Permutations can be represented by a single integer vector. In particular, if v is a reordering of the vector $0{:}n-1$, then we define the permutation P_v by

$$P_v = I_n(:, v).$$

Thus, if $v = [\,0,\ 3,\ 1,\ 2\,]$, then

$$P_v = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{bmatrix}.$$

The action of P_v on a matrix $A \in \mathbb{C}^{n\times n}$ can be neatly described in terms of v and the colon notation:

$$\begin{aligned} AP_v &= A(:, v), \\ P_v^T A &= A(v, :). \end{aligned}$$

It is easy to show that $P_v^T P_v = I_n(v,:)I_n(:,v) = I_n$ and that

$$P_v = I_n(:, v) \Rightarrow P_v^T = I_n(:, w), \quad w(v) = 0{:}n-1. \qquad (1.1.4)$$

1.1.9 Diagonal Scaling and Pointwise Multiplication

If $d \in \mathbb{C}^n$, then $D = \text{diag}(d) = \text{diag}(d_0, \ldots, d_{n-1})$ is the n-by-n diagonal matrix with diagonal entries $d_0, \ldots, d_{n-1}$. The application of D to an n-vector x is tantamount to the *pointwise multiplication* of the vector d and the vector x, which we designate by $d\,.\!*\,x$:

$$y = Dx = \text{diag}(d)x \quad \Leftrightarrow \quad y = d\,.\!*\,x \quad \Leftrightarrow \quad y_i = d_i x_i\,.$$

More generally, if $A, B \in \mathbb{C}^{m \times n}$, then the pointwise product of $(c_{kj}) = C = A\,.\!*\,B$ is defined by $c_{kj} = a_{kj}b_{kj}$. It is not hard to show that $A\,.\!*\,B = B\,.\!*\,A$ and $(A\,.\!*\,B)^H = A^H\,.\!*\,B^H$.

A few additional diagonal matrix notations are handy. If $A \in \mathbb{C}^{n \times n}$, then diag$(A)$ is the n-vector $[\, a_{00},\ a_{11},\ \ldots\ a_{n-1,n-1}\,]^T$. If $D_0, \ldots, D_{p-1}$ are matrices, then $D = \text{diag}(D_0, \ldots, D_{p-1})$ is the p-by-p block diagonal matrix defined by

$$D = \begin{bmatrix} D_0 & 0 & \cdots & 0 \\ 0 & D_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & D_{p-1} \end{bmatrix}.$$

1.1.10 Kronecker Products

The block structures that arise in FFT work are highly regular and a special notation is in order. If $A \in \mathbb{C}^{p \times q}$ and $B \in \mathbb{C}^{m \times n}$, then the *Kronecker product* $A \otimes B$ is the p-by-q block matrix

$$A \otimes B = \begin{bmatrix} a_{00}B & \cdots & a_{0,q-1}B \\ \vdots & & \vdots \\ a_{p-1,0}B & \cdots & a_{p-1,q-1}B \end{bmatrix} \in \mathbb{C}^{pm \times qn}.$$

Note that

$$I_n \otimes B = \begin{bmatrix} B & 0 & \cdots & 0 \\ 0 & B & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & B \end{bmatrix}$$

is block diagonal.

A number of Kronecker product properties are heavily used throughout the text and are listed inside the front cover for convenience. Seven Kronecker facts are required in this chapter and they are now presented. The first four results deal with products, transposes, inverses, and permutations.

Kron1

If A, B, C, and D are matrices, then

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD),$$

assuming that the ordinary multiplications AC and BD are defined.

Kron2

If A and B are nonsingular matrices, then $A \otimes B$ is nonsingular and $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$

$\boxed{\textbf{Kron3}}$

If A and B are matrices, then $(A \otimes B)^T = A^T \otimes B^T$.

$\boxed{\textbf{Kron4}}$

If P and Q are permutations, then so is $P \otimes Q$.

The first property says that the product of Kronecker products is the Kronecker product of the products. The second property says that the inverse of a Kronecker product is the Kronecker product of the inverses. The third property shows that the transpose of a Kronecker product is the Kronecker product of the transposes *in the same order*. The fourth property asserts that the Kronecker product of two permutations is a permutation.

These results are not hard to verify. For example, to establish Kron1 when all the matrices are n-by-n, define $U = A \otimes B$, $V = C \otimes D$, and $W = UV$. Let U_{kj}, V_{kj}, and W_{kj} be the kj blocks of these matrices. It follows that

$$\begin{aligned} W_{kj} &= \sum_{q=0}^{n-1} U_{kq}V_{qj} = \sum_{q=0}^{n-1}(a_{kq}B)(c_{qj}D) \\ &= \left(\sum_{q=0}^{n-1} a_{kq}c_{qj}\right) BD = [AC]_{kj}BD. \end{aligned}$$

This shows that W is the conjectured Kronecker product.

The next two Kronecker properties show that matrix-vector products of the form $z = (I \otimes A)v$ or $z = (A \otimes I)v$ are "secretly" matrix-matrix multiplications.

$\boxed{\textbf{Kron5}}$

If $A \in \mathbb{C}^{r\times r}$ and $x \in \mathbb{C}^n$ with $n = rc$, then

$$y = (I_c \otimes A)x \quad \Leftrightarrow \quad y_{r\times c} = Ax_{r\times c}.$$

$\boxed{\textbf{Kron6}}$

If $A \in \mathbb{C}^{c\times c}$ and $x \in \mathbb{C}^n$ with $n = rc$, then

$$y = (A \otimes I_r)x \quad \Leftrightarrow \quad y_{r\times c} = x_{r\times c}A^T.$$

The proofs of Kron5 and Kron6 are routine subscripting arguments. The last Kronecker property that we present in this section is a useful corollary of Kron1.

$\boxed{\textbf{Kron7}}$

If A is a matrix, then

$$I_p \otimes (I_q \otimes A) = I_{pq} \otimes A.$$

A number of other, more involved Kronecker facts are required for some Chapter 2 derivations. See §2.2.1.