DISCRETE COSINE AND SINE TRANSFORMS

General Properties, Fast Algorithms and Integer Approximations

Vladimir Britanak • Patrick Yip • Kamisetty R. Rao

Discrete Cosine and Sine Transforms

This page intentionally left blank

Discrete Cosine and Sine Transforms

General Properties, Fast Algorithms and Integer Approximations

Vladimir Britanak
Institute of Informatics
Slovak Academy of Sciences
Bratislava
Slovak Republic

Patrick C. Yip
McMaster University
Department of Mathematics and Statistics
Hamilton, Ontario
Canada

K. R. Rao
The University of Texas at Arlington
Department of Electrical Engineering
Arlington, Texas
USA

ELSEVIER

AMSTERDAM BOSTON HEIDELBERG LONDON NEW YORK OXFORD
PARIS SAN DIEGO SAN FRANCISCO SINGAPORE SYDNEY TOKYO
Academic Press is an imprint of Elsevier

Academic Press is an imprint of Elsevier
The Boulevard, Langford Lane, Kidlington, Oxford OX5 1GB, UK
84 Theobald's Road, London WC1X 8RR, UK
30 Corporate Drive, Suite 400, Burlington, MA 01803, USA
525 B Street, Suite 1900, San Diego, California 92101-4495, USA

First edition 2007

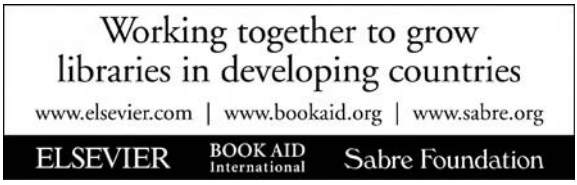
British Library Cataloguing in Publication Data
A catalogue record for this book is available from the British Library

Library of Congress Catalog Number: 2006931102

ISBN-13: 978-0-12-373624-6
ISBN-10: 0-12-373624-2

Typeset by Charon Tec Ltd (A Macmillan Company), Chennai, India
www.charontec.com
Printed and bound in Great Britain

07 08 09 10 10 9 8 7 6 5 4 3 2 1



Contents

Preface

Since the book, *Discrete Cosine Transform* by K. R. Rao and P. Yip (Academic Press, Boston) was published in 1990, the discrete cosine transform (DCT) has increasingly attracted the attention of scientific, engineering and research communities. The DCT is used in many applications and in data compression in particular. This is due to the fact that the DCT has excellent energy-packing capability and also approaches the statistically optimal Karhunen–Loéve transform (KLT) in decorrrelating a signal. The development of various fast algorithms for the efficient implementation of the DCT involving real arithmetic only, further contributed to its popularity. In the last several years there have been significant advances and developments in both theory and applications relating to transform processing of signals. In particular, digital processing motivated the investigation of other forms of DCTs for their integer approximations. International standards organizations (ISO/IEC and ITU-T) have adopted the use of various forms of the integer DCT. At the same time, the investigation of other forms of discrete sine transforms (DSTs) has made a similar impact. There is therefore a need to extend the coverage to include these techniques. This book is aimed at doing just that.

The authors have retained much of the basic theory of transforms and transform processing, since the basic mathematics remains valid and valuable. The theory and fast algorithms of the DCTs, as well as those for the DSTs, are dealt with in great detail. There is also an appendix covering some of the fundamental mathematical aspects underlying the theory of transforms. It is no exaggeration to say that applications using DCT are numerous and it is with this in mind that the authors have decided not to include applications explicitly. Readers of this book will either have practical problems requiring the use of DCT, or want to examine the more general theory and techniques for future applications. There is no practical way of comprehensively dealing with all possible applications. However, it must be emphasized that implementation of the various transforms is considered an integral part of our presentation. It is the authors' hope that readers will not only gain some understanding of the various transforms, but also take this knowledge to apply to whatever processing problems they may encounter.

The book *Discrete Cosine and Sine Transforms: General properties, Fast algorithms and Integer Approximations* is aimed at both the novice and the expert. The fervent hopes and aspirations of the authors are that the latest developments in the general DCT/DST field

further lead into additional applications and also provide the incentive and inspiration to further modify/customize these transforms with the overall motivation to improve their efficiencies while retaining the simplicity in implementations.

V. Britanak
P. C. Yip
K. R. Rao

February 2006

Acknowledgments

Many many hours and efforts have been spent behind the computer in preparing and typing electronic form of the manuscript, in analytical derivation of various matrix factorizations, drawing the figures of signal flow graphs and verifying by computer programs. This book is the result of long-term association of the three authors, V. Britanak, P. Yip and K. R. Rao. Special thanks go to their respective families for their support, perseverance and understanding. We appreciate also the continued encouragement and many helpful suggestions of our colleagues and friends in the departments of the institute and universities. Especially, the leading author wishes to thank his love, Gulinka, for her patience, understanding and encouragement during the years of preparation of this book. Finally, it is also appropriate to acknowledge here the financial support provided by Slovak Scientific Agency VEGA, Project No. 2/4149/24.

The authors have been honored to have worked with Academic Press Inc. Elsevier Science on this project. The encouragement, support and understanding for the delayed completion of the book provided by the publishing editorial staff at Materials Science Department, and in particular, Christopher Greenwell and Dr. Jonathan Agbenyega, have been greatly appreciated.

This page intentionally left blank

List of Acronyms

1-D	One-Dimensional
2-D	Two-dimensional
3-D	Three-dimensional
BinDCT	Binary arithmetic DCT
BinDST	Binary arithmetic DST
CMT	C-Matrix Transform
CPU	Central Processor Unit
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DHT	Discrete Hartley Transform
DLU	DLU matrix factorization or DLU computational structure
DPCM	Differential Pulse Code Modulation
DST	Discrete Sine Transform
DTT	Discrete Trigonometric Transform
DUL	DUL matrix factorization or DUL computational structure
DWT	Discrete W Transform
EOT	Even/Odd Transform
FCT	Fourier Cosine Transform
FFCT	Fast Fourier Cosine Transform
FFT	Fast Fourier Transform
FRDCT	Fractional Discrete Cosine Transform
FRDFT	Fractional Discrete Fourier Transform
FRDST	Fractional Discrete Sine Transform
FST	Fourier Sine Transform
GCMT	Generalized C-Matrix Transform
GCT	Generalized Chen Transform
GDFT	Generalized Discrete Fourier Transform
GDHT	Generalized Discrete Hartley Transform
HA	Half sample Antisymmetric
HS	Half sample Symmetric
ICT	Integer Cosine Transform
IEC	International Electrotechnical Commission
IEEE	Institute of Electrical and Electronics Engineers
IntDCT	Integer DCT

INTDCT	Integer DCT
IntDST	Integer DST
ISO	International Organization for Standardization
IST	Integer Sine Transform
ITU-T	International Telecommunication Standardization Sector
JPEG2000	Joint Photographic Experts Group
KLT	Karhunen–Loéve Transform
LDCT	Lossless DCT
LDST	Lossless DST
LDU	*LDU* matrix factorization
LU	*LU* matrix factorization
LUL	LUL matrix factorization or LUL computational structure
MDCT	Modified Discrete Cosine Transform
MDL	Multi-Dimensional computational structure
MDST	Modified Discrete Sine Transform
MLT	Modulated Lapped Transform
MPEG	Moving Picture Experts Group
MSE	Mean Square Error
PCA	Principal Component Analysis
PLUS	*PLUS* matrix factorization
POS	Points Of Symmetry
QR	*QR* matrix factorization
RDCT	Reversible DCT
RDST	Reversible DST
SCT	Symmetric Cosine Transform
SignDCT	Signed DCT
SMT	S-Matrix Transform
SOPOT	Sum Of Powers Of Two
SST	Symmetric Sine Transform
SVD	Singular Value Decomposition
ULD	ULD matrix factorization or ULD computational structure
ULU	ULU matrix factorization or ULU computational structure
VLSI	Very Large-Scale Integration
WA	Whole sample Antisymmetric
WHT	Walsh–Hadamard Transform
WS	Whole sample Symmetric

CHAPTER 1

Discrete Cosine and Sine Transforms

1.1 Introduction

Since the publication of original book [1] more than 15 years ago many new contributions/extensions/modifications/updates/improvements to the origin, theoretical and practical aspects of the discrete cosine transforms (DCTs) and discrete sine transforms (DSTs) have been developed. Although the original book [1] has focused almost exclusively on the fast algorithms and applications of the DCT of type II (DCT-II) which has become the heart of many established international image/video coding standards [2], since then other forms of the DCT and DST have been investigated in detail. The complete set of DCTs and DSTs, called the discrete trigonometric transforms, has found a number of digital signal processing applications. Among them, for example, the DCT/DST of type IV (DCT-IV/DST-IV) and DCT-II/DST-II are used for the efficient implementation of lapped orthogonal transforms [6] and perfect reconstruction cosine/sine modulated filter banks (known as modified discrete cosine/sine transforms (MDCTs/MDSTs) or equivalently modulated lapped transforms (MLTs) [6]) for high-quality transform/subband audio coding.

The complete set of DCTs and DSTs constituting the entire class of discrete sinusoidal unitary transforms is presented including their definitions, general mathematical properties, relations to the Karhunen–Loève transform (KLT), with the emphasis on fast algorithms and integer approximations for their efficient implementations in the integer domain. The DCTs and DSTs are real-valued transforms that map integer-valued signals to floating-point coefficients. One of the important issues for the applicability of DCTs and DSTs is the existence of fast algorithms that allow their efficient computation. Although the fast algorithms reduce the computational complexity significantly, they still need floating-point operations. To eliminate the floating-point operations, methods of integer approximations have been proposed to construct and flexibly generate a family of integer transforms with arbitrary accuracy and performance. The integer transforms currently represent the *modern transform technologies* for lossless transform-based coding. The integer DCTs/DSTs with low-cost and low-powered implementation can replace the corresponding real-valued transforms in wireless and satellite communication systems as well as portable computing applications.

The book covers various latest developments in DCTs and DSTs in a unified way, and it is essentially a detailed excursion on orthogonal/orthonormal DCT and DST matrices, their matrix factorizations and integer approximations. It is hoped that the book will serve as an excellent reference in developing integer DCTs and DSTs as well as an inspiration for further advanced research.

1.2 Organization of the book

The book is organized in terms of chapters starting with this introductory chapter; each chapter has its own list of general references and appendices.

Chapter 2 covers definitions and general properties of classical integral transforms, Fourier cosine transform and Fourier sine transform. The general properties of these continuous transforms such as inversion, linearity, shift in time/frequency, differentiation in time/frequency, asymptotic behavior, integration in time/frequency and convolution in time together with examples of integral transforms for selected continuous functions are presented in Sections 2.2–2.5. All the DCTs and DSTs are not simply discretized versions of the corresponding integral continuous transforms rather, the discretized cosine and sine functions form the basis functions for an entire family of DCTs and DSTs, and are actually eigenfunctions (or eigenvectors) of certain tridiagonal matrix forms. This issue is addressed in Sections 2.6 and 2.7. DCTs and DSTs possess nice mathematical properties such as unitarity, linearity, scaling and shift in time, and, in particular, convolution properties which are discussed in detail in Sections 2.8 and 2.9.

KLT an optimal transform from a statistical viewpoint is defined in Chapter 3 (Section 3.2) along with the demonstration of the asymptotic equivalence of DCT-I and DCT-II to KLT in Section 3.3. Section 3.4 addresses the asymptotic equivalence of different types of correlation matrices and their orthonormal representations leading to a general procedure for generating certain discrete unitary transforms for a given class of signal correlation matrices.

For the DCT and DST to be viable, feasible and practical, the fast algorithms for their efficient implementation in terms of reduced memory, implementation complexity and recursivity are essential. The fast algorithms for both one- and two-dimensional (1-D, 2-D, respectively) DCTs/DSTs are the main thrust in Chapter 4. In Section 4.2, the definitions, properties of and relations between DCTs and DSTs are first presented, followed by presentation of the explicit forms of orthonormal DCT and DST matrices for $N = 2, 4$ and 8 in Section 4.3. The fast 1-D rotation-based algorithms for the computation of DCTs and DSTs based on the (recursive) sparse matrix factorizations of the corresponding DCT and DST matrices and represented by the generalized signal flow graphs are discussed in Section 4.4. The matrix factorizations reveal various interrelations between different versions of the DCT and DST. These selected fast algorithms are very convenient in constructing integer approximations of DCTs and DSTs. Section 4.5 analyzes existing 2-D fast DCT/DST algorithms and suggests a simple method for generating 2-D direct DCT/DST algorithms from the corresponding 1-D ones.

As integer versions of the DCT/DST have attracted the attention of researchers resulting in substantial simplification in their implementation while still maintaining performance

nearly equal to their earlier versions, it is only logical that this arena be focused in much detail and depth in Chapter 5. Section 5.2 presents the basic material from linear algebra, theory of matrices and matrix computations which is fundamental for understanding the approximation methods. In order to evaluate the approximation error between the approximated and original transform matrix and to measure the performance of resulting approximated transform used in data compression, some theoretical criteria are defined in Section 5.3. Finally, various developed methods and design approaches to integer approximation of the DCT and DST are detailed in Section 5.4. More recent developments in designing lossless DCTs, invertible integer DCTs and reversible DCTs including the latest developments are discussed in Sections 5.5 and 5.6.

All chapters end with a summary, problems/exercises and references. Problems/exercises reflect the contents of the corresponding chapters and are intended for the reader in terms of refresh/review/reinforce their contents. Extensive definitions, principles, properties, signal flow graphs, derivations, proofs and examples are provided throughout the book for proper understanding of the strengths and shortcomings of the spectrum of cosine/sine transforms and their application in diverse disciplines.

1.3 Appendices

Appendices A.1 through A.3 review the important basic concepts of linear algebra such as vector spaces (Appendix A.1), matrix eigenvalue problem (Appendix A.2) and matrix decompositions (Appendix A.3) in the form of definitions and theorems with exercises/problems at the end. Deterministic as well as random signals, their classification and representations are discussed in Appendix A.4. A number of examples are listed in Appendices to illustrate the use of basic concepts in practical applications.

1.4 References

To retain the connectivity among the chapters of the book as much as possible, each chapter in the book includes its own list of references related to the discussed subject. Therefore, some references may appear in the lists of references of chapters more than once.

1.5 Additional references

An extensive list of additional references have been appended to this chapter. No claim for completeness of this list is made. Additional references, although not cited in subsequent chapters, reflect the various recent/latest developments in the efficient implementations of DCTs and DSTs, mainly 1-D, 2-D, 3-D and in general, multi-dimensional fast DCT/DST algorithms for the time period from 1989/1990 up to now. They supplement the comprehensive list of references related to DCTs and DSTs in the original books [1, 2]. Thus, this book and books [1, 2] cover completely the theoretical developments, algorithmic history of DCTs and DSTs including the recent active research topics.

For clarity, the additional references are classified into the following categories with guidelines:

- **Other books discussing DCTs, DSTs and KLT** [3–9]
  The recent published books discuss both the theoretical and practical aspects of DCTs and DSTs including the KLT.
- **Fast 1-D radix-2 DCT/DST algorithms** [10–67]
  This category is further subdivided into three parts: fast algorithms for computation of DCT-I, -II, -III, -IV and corresponding DST-I, -II, -III, -IV [10–32], fast DCT algorithms only [33–59] and fast DST algorithms only [60–67].
- **Fast direct 2-D DCT/DST algorithms** [68–87]
  This category includes the direct 2-D radix-2 DCT/DST algorithms, and direct even/prime-length 2-D algorithms based on cyclic convolutions and circular or skew-circular correlations. Since 2-D DCT/DST kernels are separable, the 2-D DCT/DST computation can simply be realized by the so-called row–column method which sequentially uses any fast 1-D DCT/DST algorithm on rows and columns of the input data matrix. In general, many 1-D DCT/DST algorithms can be extended to the direct 2-D case using a 2-D decomposition process.
- **Fast direct 3-D and multi-dimensional DCT/DST algorithms** [88–97]
  The higher-dimensional DCT/DST algorithms can be obtained by the similar methods as those of 2-D DCT/DST ones.
- **Fast even/odd/composite-length, prime-factor, radix-q and mixed-radix DCT/DST algorithms** [98–126]
  The limitation common to most fast DCT/DST algorithms is that N must be a power of 2 (radix-2 DCT/DST algorithms). In practice, various sequence lengths other than a power of 2 may occur. To deal with such sequence lengths, new fast even/odd-length (N is an even/odd integer), composite-length ($N = p \cdot q$, where p and q are relatively primes), prime-factor, radix-q ($N = q^n$, where q is an odd integer) and mixed-radix ($N = 2^n \cdot q$, where $q = 3, 5, 6, 7, 9, \ldots$) DCT/DST algorithms have been proposed. Even/odd-length and prime-factor DCT/DST algorithms can be directly mapped into the corresponding even/odd-length and prime-factor complex-valued or real-valued FFT modules, or they are based on shorter cyclic/skew-cyclic convolutions and skew-circular correlations. The algorithms for sequence lengths other than 2^n need quite different methods for their derivation, and generally they have a higher computational complexity and have more complex structure.
- **Fast pruning DCT algorithms** [127–134]
  The standard DCT (radix-2) algorithms inherently assume that the lengths of input and output data sequences are equal. However, in many applications such as data compression, the most important information about the signal is kept by the low-frequency DCT coefficients. Therefore, from N coefficients (N being the length of data sequence) only N_1 ($N_1 < N$) lowest-frequency coefficients need to be computed. Such a method where only a subset of the output coefficients is utilized to accelerate the computation is referred to as "pruning". Therefore the algorithms, called fast pruning DCT algorithms, have been developed just for this purpose. In general, the fast DCT algorithm to be pruned must be defined by a simple structured recursive matrix factorization of the transform matrix and represented by the regular signal flow graph.

- **DCT/DST computation by recursive filter structures** [135–149]
  A class of algorithms for arbitrary length forward and inverse DCT/DST computations are recursive algorithms where DCT/DST kernels are converted to regular regressive structures based on sinusoidal recursive formulae, or recurrence formulae for Chebyshev polynomials (of the second and third kind), or Clenshaw's recurrence formula. Although these recursive algorithms are not efficient in terms of computational complexity, regressive structures provide simple and efficient schemes for the parallel VLSI implementation of the variable length DCTs/DSTs.

- **Fractional DCTs and DSTs** [150–152]
  Recently, the fractional DCTs (FRDCTs) and fractional DSTs (FRDSTs) for DCT-II, symmetric cosine and symmetric sine transforms have been introduced. The definitions of FRDCTs and FRDSTs are based on eigen decompositions (eigenvalues and eigenvectors) of the corresponding DCT and DST matrices; or simply by other words, FRDCTs and FRDSTs are defined through the "fractional" real powers of DCT and DST matrices. It is the same idea as that of the fractional discrete Fourier transform (FRDFT). The investigation of FRDCT and FRDST, their general properties are recently an active and interesting research topic. Open problems involve the rigorous definitions of FRDCTs and FRDSTs for other forms of DCT and DST, study of their general properties, matrix representations and, in particular, fast algorithms for their practical implementations [152].

- **Fast quantum algorithms for DCTs and DSTs** [153]
  Quantum computing has recently become an exciting area of emerging digital signal processing applications. A classical computer does not allow to calculate N-point DCTs or DSTs, where $N = 2^n$, in less than linear time. This trivial lower bound is no longer valid for a quantum computer. In fact, it is possible to realize N-point DCTs and DSTs with as little as $O(\log_2{}^2 N)$ operations on a quantum computer, whereas the all known fast DCT/DST algorithms realized on a classical computer require $O(N \log_2 N)$ operations. Based on existing efficient quantum circuits for the DFT, the (extremely) fast quantum DCT/DST algorithms can be derived and implemented on a number of quantum computing technologies.

We believe that the additional references, although not used in the book, will be a valuable and useful source for the reader in her/his further study or advanced research or in solving specific problems in the area of DCT/DST applications.

References

[1] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, Boston MA, 1990.

[2] K. R. Rao and J. J. Hwang, *Techniques and Standards for Digital Image/Video/Audio Coding*, Prentice-Hall, Upper Saddle River, NJ, 1996.

Other books discussing DCTs, DSTs and KLT

[3] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[4] R. J. Clarke, *Transform Coding of Images*, 2nd Printing, Academic Press, London, 1990.

[5] W. K. Pratt, *Digital Image Processing*, 2nd Edition, John Wiley, New York, 1991.

[6] H. S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Norwood, MA, 1992.

[7] A. D. Poularikas, Editor, *The Transforms and Applications Handbook*, CRC & IEEE Press, Boca Raton, FL, 1996.

[8] O. K. Ersoy, *Fourier-Related Transforms, Fast Algorithms and Applications*, Prentice-Hall, Upper Saddle River, NJ, 1997.

[9] A. Mertins, *Signal Analysis: Wavelets, Filter Banks, Time–Frequency Transforms and Applications*, John Wiley & Sons, Chichester, 1999.

Fast 1-D radix-2 DCT/DST algorithms

[10] S. C. Chan and K. L. Ho, "Direct methods for computing discrete sinusoidal transforms", *IEE Proceedings – Radar and Signal Processing*, Vol. 137, Part F, No. 6, December 1990, pp. 433–442.

[11] L.-W. Chang and M.-C. Wu, "A unified systolic array for discrete cosine and sine transforms", *IEEE Transactions on Signal Processing*, Vol. 39, January 1991, pp. 192–194.

[12] R. Gluth, "Regular FFT-related transform kernels for DCT/DST-based polyphase filter banks", *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2205–2208.

[13] Z. Cvetkovič and M. V. Popovič, "New fast algorithms for the computation of discrete cosine and sine transforms", *IEEE Transactions on Signal Processing*, Vol. 40, No. 8, August 1992, pp. 2083–2086.

[14] N. Rama Murthy and M. N. S. Swamy, "On a novel decomposition of the DCT and its application", *IEEE Transactions on Signal Processing*, Vol. 41, No. 1, January 1993, pp. 480–485.

[15] O. K. Ersoy, "A comparative review of real and complex Fourier-related transforms", *Proceedings of the IEEE*, Vol. 82, No. 3, March 1994, pp. 429–447.

[16] P. S. Kumar and K. M. M. Prabhu, "A set of new fast algorithms for DCTs and DSTs", *International Journal of Electronics*, Vol. 76, No. 4, April 1994, pp. 553–568.

[17] P. Z. Lee and F.-Y. Huang, "Restructured recursive DCT and DST algorithms", *IEEE Transactions on Signal Processing*, Vol. 42, No. 7, July 1994, pp. 1600–1609.

[18] N. Rama Murthy and M. N. S. Swamy, "On the on-line computation of DCT-IV and DST-IV transforms", *IEEE Transactions on Signal Processing*, Vol. 43, No. 5, May 1995, pp. 1249–1251.

[19] J. Guo, C. Chen and C.-W. Jen, "Unified array architecture for DCT/DST and their inverses", *Electronics Letters*, Vol. 31, No. 21, 1995, pp. 1811–1812.

[20] V. Britanak, "A unified discrete cosine and discrete sine transform computation", *Signal Processing*, Vol. 43, No. 3, May 1995, pp. 333–339.

[21] C.-H. Wei and C.-F. Chen, "On the computation of the length-2^m discrete cosine and sine transforms via permuted difference coefficient", *IEEE Transactions on Signal Processing*, Vol. 44, No. 2, February 1996, pp. 387–396.

[22] W. S. Li, Z. S. Wang and Z. Y. He, "Neural network based real-time computation of DCTs and DSTs", *Electronics Letters*, Vol. 32, No. 19, September 1996, pp. 1795–1796.

[23] M. Wezelenburg, "General radix-2^n DCT and DST algorithms", *Proceedings of the International Conference on ECCTD'97*, Budapest, Hungary, September 1997, pp. 789–794.

[24] S. B. Pan and R.-H. Park, "Unified systolic array for fast computation of the discrete cosine transform, discrete sine transform and discrete Hartley transform", *Optical Engineering*, Vol. 36, No. 12, December 1997, pp. 3439–3444.

[25] S. B. Pan and R.-H. Park, "Unified systolic arrays for computation of the DCT/DST/DHT", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, 1997, pp. 413–419.

[26] J. Astola and D. Akopian, "Architecture-oriented regular algorithms for discrete sine and cosine transforms", *IEEE Transactions on Signal Processing*, Vol. 47, No. 4, April 1999, pp. 1109–1124.

[27] K. Maharatna, A. S. Dhar and S. Banerjee, "A VLSI array architecture for realization of DFT, DHT, DCT and DST", *Signal Processing*, Vol. 81, No. 9, September 2001, pp. 1813–1822.

[28] J. Takala and J. Nikara, "Unified pipeline architecture for discrete sine and cosine transforms of type IV", *Proceedings of the 3-rd International Conference on Information Communication and Signal Processing*, Singapore, October 2001.

[29] S. Poornachandra, V. Ravichandran and N. Kumaravel, "Mapping of discrete cosine transform (DCT) and discrete sine transform (DST) based on symmetries", *IETE Journal of Research*, Vol. 49, No. 1, January to February 2003, pp. 35–42.

[30] V. Kober, "Efficient algorithms for running type-I and type-III discrete sine transforms", *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, Vol. E87A, No. 3, March 2004, pp. 761–763.

[31] V. Kober, "Fast algorithms for the computation of sliding discrete sinusoidal transforms", *IEEE Transactions on Signal Processing*, Vol. 52, No. 6, June 2004, pp. 1704–1710.

[32] J. A. Nikara, J. H. Takala and J. T. Astola, "Discrete cosine and sine transforms – regular algorithms and pipeline architectures", *Signal Processing*, Vol. 86, No. 2, February 2006, pp. 230–249.

1-D DCT algorithms

[33] Y. Arai, T. Agui and M. Nakajima, "A fast DCT-SQ scheme for images", *The Transactions of the IEICE*, Vol. E 71, No. 11, November 1989, pp. 1095–1097.

[34] N. Rama Murthy and M. N. S. Swamy, "On the algorithms for the computation of even discrete cosine transform-2 (EDCT-2) of real sequences", *IEEE Transactions on Circuits and Systems*, Vol. 37, No. 5, May 1990, pp. 625–627.

[35] F. Arguello and E. L. Zapata, "Fast cosine transform based on the successive doubling method", *Electronics Letters*, Vol. 26, No. 19, September 1990, pp. 1616–1618.

[36] Z. Wang and G. A. Jullien, "Relationship between two algorithms for discrete cosine transform", *Electronics Letters*, Vol. 27, No. 3, June 1991, pp. 1114–1115.

[37] W. Li, "A new algorithm to compute the DCT and its inverse", *IEEE Transactions on Signal Processing*, Vol. 39, No. 6, June 1991, pp. 1305–1313.

[38] W. Kou and T. Fjallbrant, "A direct computation of DCT coefficients for a signal block taken from two adjacent blocks", *IEEE Transactions on Signal Processing*, Vol. 39, No. 7, July 1991, pp. 1692–1695.

[39] A. N. Skodras and A. G. Constantinides, "Efficient input-reordering algorithms for fast DCT", *Electronics Letters*, Vol. 27, No. 21, October 1991, pp. 1973–1975.

[40] J. C. Yao and C. Y. Hsu, "Fixed-point round-off error analysis for the discrete cosine transform", *International Journal of Electronics*, Vol. 72, No. 1, January 1992, pp. 45–55.

[41] Y.-H. Chan and W.-C. Siu, "A cyclic correlated structure for the realization of discrete cosine transform", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 39, No. 2, February 1992, pp. 109–113.

[42] J.-L. Wu, S.-H. Hsu and W.-J. Duh, "A novel two-stage algorithm for DCT and IDCT", *IEEE Transactions on Signal Processing*, Vol. 40, No. 6, June 1992, pp. 1610–1612.

[43] I. D. Yun and S. U. Lee, "On the fixed-point-error analysis of several fast DCT algorithms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 3, No. 1, February 1993, pp. 27–41.

[44] A. N. Skodras and C. A. Christopoulos, "Split-radix fast cosine transform algorithm", *International Journal of Electronics*, Vol. 74, No. 4, April 1993, pp. 513–522.

[45] C. Y. Hsu and J. C. Yao, "Comparative performance of fast cosine transform with fixed-point round-off error analysis", *IEEE Transactions on Signal Processing*, Vol. 42, No. 5, May 1994, pp. 1256–1259.

[46] V. Britanak, "On the computation of discrete cosine transform", *Signal Processing*, Vol. 40, No. 2–3, November 1994, pp. 183–194.

[47] N.-C. Hu and S.-W. Luoh, "Two–stage decomposition of the DCT", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 142, No. 5, October 1995, pp. 319–326.

[48] R. Storn, "Efficient input reordering for the DCT based on a real-valued decimation-in-time FFT", *IEEE Signal Processing Letters*, Vol. 3, No. 8, August 1996, pp. 242–244.

[49] Y. Jeong, I. Lee, H. S. Kim and K. T. Park, "Fast DCT algorithm with fewer multiplication stages", *Electronics Letters*, Vol. 34, No. 8, April 1998, pp. 723–724.

[50] V. Kober and G. Cristobal, "Fast recursive algorithms for short-time discrete cosine transform", *Electronics Letters*, Vol. 35, No. 15, July 1999, pp. 1236–1238.

[51] S.-F. Hsiao, W.-R. Shiue and J.-M. Tseng, "A cost-efficient and fully-pipelined architecture for DCT/IDCT", *IEEE Transactions on Consumer Electronics*, Vol. 45, No. 3, August 1999, pp. 515–525.

[52] A. N. Skodras, "Direct transform to transform computation", *IEEE Signal Processing Letters*, Vol. 6, No. 8, August 1999, pp. 202–204.

[53] J. Takala, D. Akopian, J. Astola and J. Sarinen, "Constant geometry algorithm for cosine transform", *IEEE Transactions on Signal Processing*, Vol. 48, No. 6, June 2000, pp. 1840–1843.

[54] Y. Jeong and I. Lee, "Fast discrete cosine transform structure suitable for implementation with integer computation", *Optical Engineering*, Vol. 39, No. 10, October 2000, pp. 2672–2676.

[55] J. Nikara, J. Takala, D. Akopian and J. Saarinen, "Pipeline architecture for DCT/IDCT", *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'2001)*, Vol. 4, Sydney, Australia, May 2001, pp. 902–905.

[56] Z. Guo, B. Shi and N. Wang, "Two new algorithms based on product system for discrete cosine transform", *Signal Processing*, Vol. 81, No. 9, September 2001, pp. 1899–1908.

[57] T. C. Tan, G. Bi, Y. Zeng and H. N. Tan, "DCT hardware structure for sequentially presented data", *Signal Processing*, Vol. 81, No. 11, November 2001, pp. 2333–2342.

[58] L.-P. Chau and W. C. Siu, "Efficient multiplier structure for realization of the discrete cosine transform", *Signal Processing: Image Communications*, Vol. 18, No. 7, August 2003, pp. 527–536.

[59] D. Kim and Y. Choe, "Fast algorithm for discrete cosine transform (DCT) – domain image downsampling using Winograd DCTs", *Optical Engineering*, Vol. 42, No. 9, September 2003, pp. 2485–2486.

1-D DST algorithms

[60] Z. Wang, "Fast discrete sine transform algorithms", *Signal Processing*, Vol. 19, No. 2, February 1990, pp. 91–102.

[61] A. Gupta and K. R. Rao, "A fast recursive algorithm for the discrete sine transform", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 38, No. 3, March 1990, pp. 553–557.

[62] A. Gupta and K. R. Rao, "An efficient FFT algorithm based on the discrete sine transform", *IEEE Transactions on Signal Processing*, Vol. 39, No. 2, February 1991, pp. 486–490.

[63] J. C. Yao and C.-Y. Hsu, "New approach for fast sine transform", *Electronics Letters*, Vol. 28, No. 15, July 1992, pp. 1398–1399.

[64] J. C. Yao and C.-Y. Hsu, "Comparative performance of discrete sine transforms with fixed-point round-off error analysis", *International Journal of Electronics*, Vol. 75, No. 7, August 1993, pp. 209–222.

[65] J. C. Yao and C.-Y. Hsu, "Further results on new fast recursive algorithms for the discrete cosine and sine transforms", *IEEE Transactions on Signal Processing*, Vol. 42, No. 11, November 1994, pp. 3254–3255.

[66] Z. Wang, G. A. Jullien and W. C. Miller, "Recursive algorithm for discrete sine transform with regular structure", *Electronics Letters*, Vol. 24, No. 30, November 1994, pp. 2011–2013.

[67] V. Kober, "Fast recursive algorithm from sliding discrete sine transform", *Electronics Letters*, Vol. 38, No. 25, December 2002, pp. 1747–1748.

Fast direct 2-D DCT/DST algorithms

[68] S. S. Kang and M. H. Lee, "An expanded 2-D DCT algorithm based on convolution", *IEEE Transactions on Consumer Electronics*, Vol. 39, No. 3, August 1993, pp. 159–165.

[69] V. Britanak, "A generalized signal flow graph for the 2-D DCT computation", *Applied Signal Processing*, Vol. 1, No. 2, February 1994, pp. 76–93.

[70] C. L. Wang and C.-Y. Chen, "High-throughput VLSI architectures for the 1-D and 2-D discrete cosine transform", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, 1995, pp. 31–40.

[71] Y. T. Chang and C. L. Wu, "New systolic array implementation of the 2-D discrete cosine transform and its inverse", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 5, 1995, pp. 150–157.

[72] Y.-P. Lee, T.-H. Chen, L.-G. Chen, M.-J. Chen and C. W. Ku, "A cost-effective architecture for 8×8 two-dimensional DCT/IDCT using direct method", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 7, No. 3, June 1997, pp. 459–467.

[73] H. R. Wu and Z. H. Mon, "Comment on fast algorithms and implementations of 2-D DCT", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 2, 1998, pp. 128–129.

[74] F. Arguello, M. A. Trenas, J. Lopez and E. L. Zapata, "Algorithm for cosine transform of Toeplitz matrices", *Electronics Letters*, Vol. 34, No. 12, June 1998, pp. 1182–1183.

[75] W.-H. Fang, N.-C. Hu and S.-K. Shih, "Recursive fast computation of the two-dimensional discrete cosine transform", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 146, No. 1, February 1999, pp. 25–33.

[76] J. Kwak and J. You, "One- and two-dimensional constant geometry fast cosine transform algorithms and architectures", *IEEE Transactions on Signal Processing*, Vol. 47, No. 7, July 1999, pp. 2023–2034.

[77] G. Bi, G. Li, K.-K. Ma and T. C. Tan, "On the computation of two-dimensional DCT", *IEEE Transactions on Signal Processing*, Vol. 48, No. 4, April 2000, pp. 1171–1183.

[78] T. C. Tan, G. Bi and H. N. Tan, "Fast recursive algorithms for 2-D discrete cosine transform", *Signal Processing*, Vol. 80, No. 9, September 2000, pp. 1917–1935.

[79] H. Lim, V. Piuri and E. E. Swartzlander Jr., "A serial–parallel architecture for two-dimensional discrete cosine and inverse discrete cosine transforms", *IEEE Transactions on Computers*, Vol. 49, No. 12, December 2000, pp. 1297–1309.

[80] J. Nikara, J. Takala, "Unified pipeline architecture for in-order 8×8 DCT and IDCT", *Proceedings of the 5th World Multiconference on Systemics, Cybernetics and Informatics*, Vol. 4, Orlando, FL, July 2001, pp. 30–35.

[81] L. Z. Cheng, "On computing the two-dimensional (2-D) type IV discrete cosine transform (2-D DCT-IV)", *IEEE Signal Processing Letters*, Vol. 8, No. 8, August 2001, pp. 239–241.

[82] S.-F. Hsiao and W.-R. Shiue, "A new hardware-efficient algorithm and architecture for computation of 2-D DCTs on a linear array", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 11, November 2001, pp. 1149–1159.

[83] S.-F. Hsiao and J.-M. Tseng, "New matrix formulation for two-dimensional DCT/IDCT computation and its distributed-memory VLSI implementation", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 149, No. 2, April 2002, pp. 97–107.

[84] J. Takala, J. Nikara and K. Punkka, "Pipeline architecture for two-dimensional discrete cosine transform and its inverse", *Proceedings of the 9th International Conference on Electronics Circuits and Systems*, Vol. 3, Dubrovnik, Croatia, September 2002, pp. 947–950.

[85] Y.-J. Chuang and J.-L. Wu, "Direct splitting and merging of 2-D DCT in the DCT domain", *Digital Signal Processing*, Vol. 14, No. 6, November 2004, pp. 614–624.

[86] Y.-J. Chuang and J.-L. Wu, "An efficient algorithm for splitting an 8×8 DCT into four 4×4 modified DCTs used in AVC/H.264", *Proceedings of the 5th EURASIP Conference focused*

on Speech and Image Processing (EC-SIP-M'2005), Smolenice, Slovak Republic, June-July 2005, pp. 311–316.

[87] C. P. Fan, "Fast algorithm designs for low-complexity 4×4 discrete cosine transform", *IEICE Transactions on Fundamentals of Electronics Communications and Computer Sciences*, Vol. E88A, No. 11, November 2005, pp. 3225–3229.

Fast direct 3-D and multi-dimensional DCT/DST algorithms

[88] Z. Wang, Z. He, C. Zou and J. D. Z. Chen, "A generalized fast algorithm for n-D discrete cosine transform and its application to motion picture coding", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 46, No. 5, May 1999, pp. 617–627.

[89] A. Elnaggar and H. M. Alnuweiri, "A new multidimensional recursive architecture for computing the discrete cosine transform", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 10, No. 1, February 2000, pp. 113–119.

[90] Y. Zeng, G. Bi and A. R. Leyman, "New polynomial transform algorithm for multidimensional DCT", *IEEE Transactions on Signal Processing*, Vol. 48, No. 10, October 2000, pp. 2814–2821.

[91] Y. Zeng, G. Bi and A. C. Kot, "New algorithm for multidimensional type-III DCT", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 47, No. 12, December 2000, pp. 1523–1529.

[92] L. Z. Cheng and Y. H. Zeng, "New fast algorithm for multidimensional type-IV DCT", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 148, No. 4, Aug. 2001, pp. 263–268.

[93] G. Bi, Y. Zeng and Y. Q. Chen, "Prime factor algorithm for multidimensional discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 49, No. 9, September 2001, pp. 2156–2161.

[94] L. Z. Cheng, Z. Wang and Z. H. Zhang, "Fast unified computation of the multidimensional discrete sinusoidal transforms", *Applied Mathematics and Computation*, Vol. 132, No. 2–3, November 2002, pp. 455–487.

[95] Y. Zeng, G. Bi and Z. P. Lin, "Combined polynomial transform and radix-q algorithm for multi-dimensional DCT-III", *Multidimensional Systems and Signal Processing*, Vol. 13, No. 1, January 2002, pp. 79–99.

[96] Y. Zeng, Z. Lin, G. Bi and L. Cheng, "Fast computation of MD-DCT-IV/MD-DST-IV by MD-DWT or MD-DCT-II", *SIAM Journal of Scientific Computing*, Vol. 24, No. 6, 2003, pp. 1903–1918.

[97] S. Boussakta and H. O. Alshibami, "Fast algorithm for the 3-D DCT-II", *IEEE Transactions on Signal Processing*, Vol. 52, No. 4, April 2004, pp. 992–1001.

Fast even/odd/composite-length, prime-factor, radix-q and mixed-radix DCT/DST algorithms

[98] S.-C. Chan and W.-C. Siu, "Efficient index mapping for computing discrete cosine transform", *Electronics Letters*, Vol. 25, No. 22, October 1989, pp. 1499–1500.

[99] N. I. Cho and S. U. Lee, "DCT algorithms for VLSI parallel implementations", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 38, No. 1, January 1990, pp. 121–127.

[100] Y.-H. Chan and W.-C. Siu, "Algorithm for prime length discrete cosine transform", *Electronics Letters*, Vol. 26, No. 3, February 1990, pp. 206–208.

[101] C. Chakrabarti and J. JaJa, "Systolic architectures for the computation of the discrete Hartley and the discrete cosine transforms based on prime factor decomposition", *IEEE Transactions on Computers*, Vol. 39, No. 11, November 1990, pp. 1359–1368.

[102] M. T. Heideman, "Computation of an odd-length DCT from real-valued DFT of the same size", *IEEE Transactions on Signal Processing*, Vol. 40, No. 1, January 1992, pp. 54–61.

[103] S.-C. Chow and K.-L. Ho, "Fast algorithms for computing the discrete cosine transform", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 39, No. 3, March 1992, pp. 185–190.

[104] Y.-H. Chan and W.-C. Siu, "Fast radix-3/6 algorithms for the realization of the discrete cosine transform", *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS'92)*, Vol. 1, San Diego, CA, May 1992, pp. 153–156.

[105] J. Guo, C. M. Liu and C.-W. Jen, "A new array architecture for prime-length discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 41, No. 1, January 1993, pp. 436–442.

[106] Y.-H. Chan and W.-C. Siu, "Mixed-radix discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 41, No. 11, November 1993, pp. 3157–3161.

[107] P. Z. Lee and F.-Y. Huang, "An efficient prime-factor algorithm for the discrete cosine transform and its hardware implementations", *IEEE Transactions on Signal Processing*, Vol. 42, No. 8, November 1994, pp. 1996–2005.

[108] D. C. Kar and V. V. B. Rao, "On the prime factor decomposition algorithm for the discrete sine transform", *IEEE Transactions on Signal Processing*, Vol. 42, No. 11, November 1994, pp. 3258–3260.

[109] E. Feig and E. Linzer, "Scaled DCTs on input sizes that are composite", *IEEE Transactions on Signal Processing*, Vol. 43, No. 1, January 1995, pp. 43–50.

[110] A. Tatsaki, C. Dre, T. Stourakis and C. Goutis, "On the computation of the prime factor DST", *Signal Processing*, Vol. 42, No. 3, March 1995, pp. 231–236.

[111] A. Tatsaki, C. Dre, T. Stourakis and C. Goutis, "Prime factor DCT algorithms", *IEEE Transactions on Signal Processing*, Vol. 43, No. 3, March 1995, pp. 772–776.

[112] A. N. Skodras and A. G. Constantinides, "Radix-3 fast discrete cosine transform algorithm", *Proceedings of the International Conference on Digital Signal Processing (DSP'95)*, Limassol, Cyprus, June 1995, pp. 819–824.

[113] N.-C. Hu and K.-C. Lin, "Skew-circular/circular correlation decomposition of prime-factor DCT", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 142, No. 4, August 1995, pp. 241–246.

[114] Y.-T. Chang and C.-L. Wang, "A new fast DCT algorithm and its systolic VLSI implementation", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 44, No. 11, November 1997, pp. 959–962.

[115] G. Bi and L. W. Yu, "DCT algorithms for composite sequence lengths", *IEEE Transactions on Signal Processing*, Vol. 46, No. 3, March 1998, pp. 554–562.

[116] J. G. Liu, H. F. Li, F. H. Y. Chan and F. K. Lam, "Fast discrete cosine transform via computation of moments", *Journal of VLSI Signal Processing Systems for Signal Image and Video Technology*, Vol. 19, No. 3, August 1998, pp. 257–268.

[117] G. Bi, "Index mapping for prime factor algorithm of discrete cosine transform", *Electronics Letters*, Vol. 35, No. 3, February 1999, pp. 198–200.

[118] G. Bi, "Fast algorithms for type-III DCT of composite sequence lengths", *IEEE Transactions on Signal Processing*, Vol. 47, No. 7, July 1999, pp. 2053–2059.

[119] J. G. Liu, F. H. Y. Chan, F. K. Lam and H. F. Li, "Moment-based fast discrete sine transforms", *IEEE Signal Processing Letters*, Vol. 7, No. 8, August 2000, pp. 227–229.

[120] G. Kutz and H. Ur, "Improved DCT-DST prime factor algorithms", *Signal Processing*, Vol. 81, No. 2, February 2001, pp. 335–343.

[121] R.-X. Yin and W.-C. Siu, "A new fast algorithm for computing prime-length DCT through cyclic convolutions", *Signal Processing*, Vol. 81, No. 5, May 2001, pp. 895–906.

[122] J.-I. Guo and C.-C. Li, "A generalized architecture for the one-dimensional discrete cosine and sine transforms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 11, No. 7, July 2001, pp. 874–881.

[123] L.-P. Chau, D. P.-K. Lun and W.-C. Siu, "Efficient prime factor algorithm and address generation techniques for the discrete cosine transform", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 48, No. 10, October 2001, pp. 985–988.

[124] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad and T. Stouraitis, "A systolic array architecture for the discrete sine transform", *IEEE Transactions on Signal Processing*, Vol. 50, No. 9, September 2002, pp. 2347–2354.

[125] S.-C. Pei and M.-P. Kao, "Direct N-point DCT computation from three adjacent $N/3$-point DC coefficients", *IEEE Signal Processing Letters*, Vol. 89, No. 2, February 2005, pp. 89–92.

[126] D. F. Chiper, M. N. S. Swamy, M. O. Ahmad and T. Stouraitis, "Systolic algorithms and a memory-based design approach for a unified architecture for the computation of DCT/DST/IDCT/IDST", *IEEE Transactions on Circuits and Systems – I: Regular Papers*, Vol. 52, No. 6, June 2005, pp. 1125–1137.

Fast pruning DCT algorithms

[127] Z. Wang, "Pruning the fast discrete cosine transform", *IEEE Transactions on Communications*, Vol. 39, No. 5, May 1991, pp. 640–643.

[128] A. N. Skodras, "Fast discrete cosine transform pruning", *IEEE Transactions on Signal Processing*, Vol. 42, No. 7, July 1994, pp. 1833–1837.

[129] C. A. Christopoulos, J. Bormans, J. Cornelis and A. N. Skodras, "The vector–radix fast cosine transform: pruning and complexity analysis", *Signal Processing*, Vol. 43, No. 2, May 1995, pp. 197–205.

[130] M. El-Sharkawy and W. Eshmawy, "A fast recursive pruned discrete cosine transform", *Digital Signal Processing*, Vol. 5, No. 3, July 1995, pp. 140–148.

[131] K.-T. Lo and W.-K. Cham, "Analysis of pruning fast cosine transform", *IEEE Transactions on Signal Processing*, Vol. 44, No. 3, March 1996, pp. 714–717.

[132] M. El-Sharkawy and W. Eshmawy, "A fast pruned 8×8 DCT algorithm", *Digital Signal Processing*, Vol. 6, No. 3, July 1996, pp. 145–154.

[133] K.-L. Chung and W.-M. Yan, "On matrix factorizations for recursive pruned discrete cosine transforms", *Signal Processing*, Vol. 68, No. 2, July 1998, pp. 175–182.

[134] S. Venkatesh and S. Srinivasan, "Modified butterfly structure for efficient implementation of pruned fast cosine transform", *Electronics Letters*, Vol. 34, No. 14, July 1998, pp. 1383–1385.

DCT/DST computation by recursive filter structures

[135] J. Canaris, "A VLSI architecture for real time computation of discrete trigonometric transforms", *Journal of VLSI Signal Processing*, Vol. 5, 1993, pp. 95–104.

[136] K. J. R. Liu and C. T. Chiu, "Unified parallel lattice structures for time-recursive discrete cosine/sine/Hartley transforms", *IEEE Transactions on Signal Processing*, Vol. 41, No. 3, March 1993, pp. 1357–1377.

[137] K. J. R. Liu, C. T. Chiu, R. K. Kolagotla and J. F. JaJa, "Optimal unified architectures for the real-time computation of time-recursive discrete sinusoidal transforms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, No. 3, March 1994, pp. 168–180.

[138] L.-P. Chau and W.-C. Siu, "Recursive algorithm for the discrete cosine transform with general lengths", *Electronics Letters*, Vol. 30, No. 3, February 1994, pp. 197–198 (Errata, *Electronics Letters*, Vol. 30, No. 7, March 1994, p. 608).

[139] L.-P. Chau and W.-C. Siu, "Efficient formulation for the realization of discrete cosine transform using recursive structure", *Proceedings of the IEEE ICASSP'94*, Vol. 3, Adelaide, Australia, April 1994, pp. III.437-III.440.

[140] Z. Wang, G. A. Jullien and W. C. Miller, "Recursive algorithms for the forward and inverse discrete cosine transform with arbitrary length", *IEEE Signal Processing Letters*, Vol. 1, No. 7, July 1994, pp. 101–102.

[141] Y.-H. Chan, L.-P. Chau and W.-C. Siu, "Efficient implementation of the discrete cosine transform using recursive filter structure", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 4, No. 6, December 1994, pp. 550–552.

[142] M. F. Aburdene, J. Zheng and R. J. Kozick, "Computation of discrete cosine transform using Clenshaw's recurrence formula", *IEEE Signal Processing Letters*, Vol. 2, No. 8, August 1995, pp. 155–156.

[143] L.-P. Chau and W.-C. Siu, "Transform domain recursive algorithm to compute discrete cosine and sine transforms", *International Journal of Electronics*, Vol. 80, No. 3, 1996, pp. 433–439.

[144] H.-C. Chang and J.-C. Liu, "A regressive structure for on-line computation of arbitrary length DCT-IV and DST-IV transforms", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 6, No. 6, December 1996, pp. 692–695.

[145] D.-Y. Chan, J.-F. Yang and S.-Y. Chen, "Regular implementation algorithms of time domain aliasing cancellation", *IEE Proceedings – Vision Image and Signal Processing*, Vol. 143, No. 6, December 1996, pp. 387–392.

[146] J.-F. Yang and C.-P. Fan, "Recursive discrete cosine transforms with selectable fixed-coefficient filters", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 46, No. 2, February 1999, pp. 211–216.

[147] J.-L. Wang, C.-B. Wu, B.-D. Liu and J.-F. Yang, "Implementation of the discrete cosine transform and its inverse by recursive structures", *Proceedings of the IEEE Workshop on Signal Processing Systems, Design and Implementation (SiPS'99)*, Taipei, Taiwan, October 1999, pp. 120–130.

[148] J.-F. Yang and C.-P. Fan, "Compact recursive structures for discrete cosine transform", *IEEE Transactions on Circuits and Systems – II: Analog and Digital Signal Processing*, Vol. 47, No. 4, April 2000, pp. 314–321.

[149] L.-P. Chau and W.-C. Siu, "Efficient recursive algorithm for the inverse discrete cosine transform", *IEEE Signal Processing Letters*, Vol. 7, No. 10, October 2000, pp. 276–277.

Fractional DCTs and DSTs

[150] S.-C. Pei and M.-H. Yeh, "The discrete fractional cosine and sine transforms", *IEEE Transactions on Signal Processing*, Vol. 49, No. 6, June 2001, pp. 1198–1207.

[151] C.-C. Tseng, "Eigenvalues and eigenvectors of generalized DFT, generalized DHT, DCT-IV and DST-IV matrices", *IEEE Transactions on Signal Processing*, Vol. 50, No. 4, April 2002, pp. 866–877.

[152] G. Cariolaro, T. Erseghe and P. Kraniauskas, "The fractional discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 50, No. 4, April 2002, pp. 902–911.

Fast quantum algorithms for DCTs and DSTs

[153] A. Klappenecker and M. Rotteler, "Discrete Cosine Transforms on quantum computers", *Proceedings of the 2nd International Symposium on Image and Signal Processing and Analysis*, Pula, Croatia, June 2001, pp. 464–468.

CHAPTER 2

Definitions and General Properties

2.1 Introduction

Transforms, and in particular integral transforms, are used primarily for the reduction of complexity in mathematical problems. Differential equations and integral equations may, by judicious application of appropriate transforms, be changed into algebraic equations, whose solutions are more easily obtained. It is thus important to derive the basic mathematical properties of these transforms before applications are considered. Transform analysis, as applied in digital signal processing, bears a similar aim. The Fourier transform, which decomposes a signal into its frequency components, and the Karhunen–Loéve transform (KLT), which decorrelates a signal sequence, are well-known examples in the digital signal processing area. Here the mathematical properties are also important.

In discussing the discrete cosine transform (DCT) and the discrete sine transform (DST), we shall first consider the continuous versions of these, i.e., the Fourier cosine transform (FCT) and the Fourier sine transform (FST). The properties of these continuous transforms are well known and bear great resemblance to those of DCT and DST. It is tempting to treat DCT and DST as discretized approximations of the continuous transforms. This would be quite mistaken. As pointed out so elegantly by Strang [1] the family of DCTs and DSTs is a natural outcome of different combinations of homogeneous boundary conditions applied to the discretized solution of a simple harmonic oscillator equation. Hence, while cosine and sine functions are the eigenfunctions of the homogeneous harmonic oscillator system, the discretized cosine and sine functions, which form the basis functions for the family of DCTs and DSTs, are eigenfunctions (or eigenvectors) in the discretized or matrix version of the homogeneous harmonic oscillator system. This connection of the DCT and DST to the simple harmonic oscillator is both elegant and amazing. The discretization of the simple harmonic oscillator system reflects the reality in which one has to deal with samples, measurements and time instants, all of which are discrete in nature. The notion of a continuum may be regarded as simply an idealization to permit the use of the calculus. On the other hand, the calculus is a powerful tool that has brought forward many important results. The properties of FCT and FST are such results. As a convenient and reasonable

reference point, we shall start with the definitions and properties of the FCT and FST in the following sections.

2.2 The FCT

We start by recalling the definition of the Fourier transform. Given a function $x(t)$ for $-\infty < t < \infty$, its Fourier transform is given by (e.g., see Elliott and Rao [2], Sneddon [3] or Poularikas [4]):

$$X(\omega) \equiv F[x(t)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} x(t) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t \tag{2.1a}$$

subject to the usual existence conditions for the integral. Here, $\mathrm{j} = \sqrt{-1}$, and $\omega = 2\pi f$ is the radian frequency and f is the frequency in Hertz. The function $x(t)$ can be recovered by the inverse Fourier transform, i.e.,

$$x(t) \equiv F^{-1}[X(\omega)] = \left(\frac{1}{2\pi}\right)^{1/2} \int_{-\infty}^{\infty} X(\omega) \mathrm{e}^{\mathrm{j}\omega t} \mathrm{d}\omega. \tag{2.1b}$$

In (2.1), $F[\cdot]$ and $F^{-1}[\cdot]$ denote respectively the forward and the inverse Fourier transforms of the functions enclosed. It is important to note here that the definitions used for the forward and inverse Fourier transforms are symmetric in the scale factor $(1/2\pi)^{1/2}$. Other conventions include having a unit scale factor for the forward transform and $(1/2\pi)$ for the inverse transform. One has to exercise care in using tables of Fourier transform properties. These properties are dependent on the definitions used. Similar care must also be exercised for properties of FCT and FST. If $x(t)$ is defined only for $t \geq 0$, we can construct a function $y(t)$ given by

$$\begin{aligned} y(t) &= x(t) \qquad t \geq 0, \\ &= x(-t) \quad t \leq 0. \end{aligned}$$

Then,

$$\begin{aligned} F[y(t)] &= \left(\frac{1}{2\pi}\right)^{1/2} \left\{ \int_0^{\infty} x(t) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t + \int_{-\infty}^{0} x(-t) \mathrm{e}^{-\mathrm{j}\omega t} \mathrm{d}t \right\} \\ &= \left(\frac{1}{2\pi}\right)^{1/2} \int_0^{\infty} x(t) [\mathrm{e}^{-\mathrm{j}\omega t} + \mathrm{e}^{\mathrm{j}\omega t}] \, \mathrm{d}t \\ &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^{\infty} x(t) \cos(\omega t) \, \mathrm{d}t. \end{aligned} \tag{2.2}$$

We can now define this as the FCT of $x(t)$ given by

$$X_c(\omega) \equiv F_c[x(t)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)\,dt. \tag{2.3}$$

Noting that $X_c(\omega)$ is an even function of ω, we can apply the Fourier inversion to (2.2) to obtain

$$y(t) = x(t) \equiv F_c^{-1}[X_c(\omega)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty X_c(\omega)\cos(\omega t)\,d\omega, \quad t \geq 0. \tag{2.4}$$

Equations (2.3) and (2.4) define a FCT pair. Some of the properties are immediately obvious:

(a) *Inversion*:

$$F_c = F_c^{-1} \tag{2.5}$$

It is clear from (2.3) and (2.4) that

$$F_c\{F_c[x(t)]\} = x(t), \quad t \geq 0.$$

(b) *Linearity*:

$$F_c[\alpha x(t) + \beta y(t)] = \alpha X_c(\omega) + \beta Y_c(\omega), \tag{2.6}$$

where α and β are constants. F_c is clearly a linear operator.

(c) *Scaling in time*:

$$\begin{aligned} F_c[x(at)] &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(at)\cos(\omega t)\,dt \\ &= a^{-1}\left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(\tau)\cos\left(\frac{\omega\tau}{a}\right)d\tau, \quad \tau = at, \\ &= a^{-1}X_c\left(\frac{\omega}{a}\right), \quad \text{for real } a > 0. \end{aligned} \tag{2.7}$$

Note the inverse scaling in the frequency domain.

(d) *Shift in time*:

$$F_c[x(t-\alpha)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t-\alpha)\cos(\omega t)\,dt$$

$$= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(\tau)\cos[\omega(\tau+\alpha)]\,d\tau, \quad \tau = t-\alpha,$$

$$= \cos(\omega\alpha)\,F_c[x(t)] - \sin(\omega\alpha)\,F_s[x(t)], \tag{2.8a}$$

where F_s denotes the FST given by

$$F_s[x(t)] \equiv \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\sin(\omega t)\,dt. \tag{2.8b}$$

We have also made the assumption that $x(t)$ vanishes for negative t.

(e) *Shift in frequency*:

$$X_c(\omega-\beta) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos[(\omega-\beta)t]\,dt, \quad \text{for } \beta > 0,$$

$$= \left(\frac{2}{\pi}\right)^{1/2} \left[\int_0^\infty x(t)\cos(\beta t)\cos(\omega t)\,dt + \int_0^\infty x(t)\sin(\beta t)\sin(\omega t)\,dt\right]$$

$$= F_c[x(t)\cos(\beta t)] + F_s[x(t)\sin(\beta t)]. \tag{2.9a}$$

Similarly, it can be shown that

$$X_c(\omega+\beta) = F_c[x(t)\cos(\beta t)] - F_s[x(t)\sin(\beta t)]. \tag{2.9b}$$

Combining (2.9a) and (2.9b) provides a *shift in frequency* result involving only FCT:

$$F_c[x(t)\cos(\beta t)] = \frac{1}{2}[X_c(\omega-\beta) + X_c(\omega+\beta)]. \tag{2.9c}$$

We note here that very similar result for a *shift in time* property can be derived from (2.8b) involving only FCT.

(f) *Differentiation in time*:

$$F_c\left[\frac{d}{dt}x(t)\right] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty \left[\frac{d}{dt}x(t)\right] \cos(\omega t)\, dt$$

$$= \left(\frac{2}{\pi}\right)^{1/2} \left\{ [x(t)\cos(\omega t)]|_0^\infty + \omega \int_0^\infty x(t)\sin(\omega t)\, dt \right\}$$

$$= -\left(\frac{2}{\pi}\right)^{1/2} x(0) + \omega F_s[x(t)]. \tag{2.10}$$

We have assumed that $x(t)$ vanishes as t tends to infinity and that the function is completely continuous and differentiable. Transforms of higher derivatives may be obtained in a similar fashion. In addition, transforms of even-order derivatives will involve only FCT.

(g) *Differentiation in frequency*:
Similar to differentiation in the time domain, the transform operation reduces a differentiation operation into multiplication by an appropriate power of the conjugate variable. For the second-order derivative, we have

$$X_c^{(2)}(\omega) = \frac{d^2}{d\omega^2}\left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)\, dt$$

$$= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)(-1)(t^2)\cos(\omega t)\, dt$$

$$= F_c[(-1)t^2 x(t)]. \tag{2.11a}$$

In general, for even-order derivatives, we have

$$X_c^{(2n)} = F_c[(-1)^n t^{2n} x(t)]. \tag{2.11b}$$

For odd orders, the FST results

$$X_c^{(2n+1)} = F_s[(-1)^{n+1} t^{2n+1} x(t)]. \tag{2.11c}$$

(h) *Asymptotic behavior*:
When the function $x(t)$ is piecewise continuous and absolutely integrable over the region $[0, \infty)$, it can be shown that for the cosine transform that

$$\lim_{\omega \to \infty} X_c(\omega) = 0. \tag{2.12}$$

The result is based on the Riemann–Lebesque theorem.

(i) *Integration in time*:

$$F_c\left[\int_t^\infty x(\tau)\mathrm{d}\tau\right] = \left(\frac{2}{\pi}\right)^{1/2}\int_0^\infty\int_t^\infty x(\tau)\,\mathrm{d}\tau\cos(\omega t)\,\mathrm{d}t$$
$$= \left(\frac{2}{\pi}\right)^{1/2}\int_0^\infty\left[\int_0^\tau \cos(\omega t)\,\mathrm{d}t\right]x(\tau)\,\mathrm{d}\tau$$

by reversing the order of the integration. Hence, we have

$$F_c\left[\int_t^\infty x(\tau)\,\mathrm{d}\tau\right] = \frac{1}{\omega}F_s[x(t)]. \tag{2.13}$$

(j) *Integration in frequency*:
A similar and almost symmetric result exists for the integration in frequency,

$$\int_\omega^\infty X_c(\beta)\,\mathrm{d}\beta = F_s\left[-\frac{1}{t}x(t)\right]. \tag{2.14}$$

In (2.14) property (h) has been invoked.

(k) *Convolution in time*:
In order to use the convolution theorem for the Fourier transform to derive the same theorem for the FCT, the function defined over the positive real line has to be extended over the entire real line. This can be done using an even extension. Let $x(t)$ and $y(t)$ be functions defined over $[0, \infty)$ and let their FCT be denoted by X_c and Y_c, respectively. Define the even extensions of these functions by

$$x_e(t) = x(|t|) \quad \text{and} \quad y_e(t) = y(|t|).$$

Then the convolution of x_e and y_e is given by

$$x_e * y_e = \int_{-\infty}^\infty x_e(\tau)y_e(t-\tau)\,\mathrm{d}\tau, \tag{2.15}$$

where $*$ denotes the convolution operation. Using the definition for the even extension, it is not difficult to see that in terms of the original functions (2.15) can be written as

$$x_e * y_e = \int_0^\infty x(\tau)[y(|t-\tau|) + y(t+\tau)]\,\mathrm{d}\tau,$$

which is easily seen as an even function of t. Applying the Fourier transform operator on both sides and using the Fourier convolution theorem, we obtain the convolution theorem for the FCT,

$$2\pi X_{\mathrm{c}}(\omega)Y_{\mathrm{c}}(\omega) = F_{\mathrm{c}}\left\{\int_0^\infty x(\tau)[y(t+\tau)+y(|t-\tau|)]\,\mathrm{d}\tau\right\}. \tag{2.16}$$

This result is clearly not as elegant as the corresponding result for the Fourier transform. The consequence is also felt in the study of DCTs.

2.3 Some examples of the FCT

(a) *The unit rectangular pulse*:

$$x(t) = U(t) - U(t-1), \quad \text{where } U(t) = \begin{cases} 0 & \text{for } t < 0, \\ 1 & \text{for } t > 0, \end{cases}$$

is the Heaviside unit step function. Its FCT is given by

$$X_{\mathrm{c}}(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^1 \cos(\omega t)\,\mathrm{d}t = \left(\frac{2}{\pi}\right)^{1/2} \frac{\sin(\omega)}{\omega}. \tag{2.17}$$

This is a well-known result producing the *sinc* function.

(b) *The inverse quadratic function*:

$$x(t) = (\alpha^2 + t^2)^{-1}, \quad \mathrm{Re}(\alpha) > 0.$$

Its FCT is obtained using a properly chosen contour integration,

$$X_{\mathrm{c}}(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\cos(\omega t)\,\mathrm{d}t = \left(\frac{\pi}{2}\right)^{1/2} \frac{\mathrm{e}^{-\alpha\omega}}{\alpha}. \tag{2.18}$$

(c) *The exponential function*:

$$x(t) = \mathrm{e}^{-\alpha t}, \quad \mathrm{Re}(\alpha) > 0.$$

Except for a scaling factor, its FCT is exactly the same as the Laplace transform of the cosine function,

$$X_{\mathrm{c}}(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty \mathrm{e}^{-\alpha t}\cos(\omega t)\,\mathrm{d}t = \left(\frac{2}{\pi}\right)^{1/2} \frac{\alpha}{\alpha^2+\omega^2}. \tag{2.19}$$

(d) *The sinc function*:

$$x(t) = \frac{\sin(at)}{t}, \quad a > 0.$$

As can be expected, its FCT behaves much like a step function,

$$\begin{aligned} X_c(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty \frac{\sin(at)}{t} \cos(\omega t)\, dt &= \left(\frac{\pi}{2}\right)^{1/2}, && \text{if } \omega < a, \\ &= \frac{1}{2}\left(\frac{\pi}{2}\right)^{1/2}, && \text{if } \omega = a, \\ &= 0, && \text{otherwise.} \end{aligned} \tag{2.20}$$

(e) *The decaying sine function*:

$$x(t) = e^{-\beta t} \sin(at), \quad a,\ \mathrm{Re}(\beta) > 0.$$

The result of the FCT can be easily understood as related to the Laplace transform of the function $\sin(at)\cos(\omega t)$ as can be seen here,

$$\begin{aligned} X_c(\omega) &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty e^{-\beta t} \sin(at)\cos(\omega t)\, dt \\ &= (2\pi)^{-\frac{1}{2}} \left[\frac{a+\omega}{\beta^2 + (a+\omega)^2} + \frac{a-\omega}{\beta^2 + (a-\omega)^2}\right]. \end{aligned} \tag{2.21}$$

Very similar result is obtained for the decaying cosine function.

(f) *Bessel function of the first kind*:
$x(t) = J_0(at)$, where $a > 0$ and J_0 denotes the zero-order Bessel function of the first kind. Its FCT is given by

$$\begin{aligned} X_c(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty J_0(at)\cos(\omega t)\, dt &= \left(\frac{2}{\pi}\right)^{1/2} (a^2 - \omega^2)^{-1/2}, && \text{for } 0 < \omega < a, \\ &= \infty, && \text{for } \omega = a, \\ &= 0, && \text{for } \omega > a. \end{aligned} \tag{2.22}$$

2.4 The FST

For the FST of a function $x(t)$, $t \geq 0$, we consider its odd extension and define the function

$$\begin{aligned} y(t) &= x(t) \qquad t \geq 0, \\ &= -x(-t) \quad t \leq 0. \end{aligned}$$

Applying the Fourier transform operator to this odd extension of the function $x(t)$ results in

$$\begin{aligned} Y(\omega) = F[y(t)] &= (2\pi)^{-1/2} \left\{ \int_0^\infty x(t)\mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t - \int_{-\infty}^0 x(-t)\mathrm{e}^{-\mathrm{j}\omega t}\,\mathrm{d}t \right\} \\ &= (2\pi)^{-1/2} \int_0^\infty x(t)[\mathrm{e}^{-\mathrm{j}\omega t} - \mathrm{e}^{\mathrm{j}\omega t}]\,\mathrm{d}t \\ &= -\mathrm{j}\left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\sin(\omega t)\,\mathrm{d}t. \end{aligned} \tag{2.23}$$

Taking the negative imaginary part of this function yields the FST of $x(t)$, or more directly,

$$X_\mathrm{s}(\omega) \equiv F_\mathrm{s}[x(t)] = \mathrm{j}F[y(t)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\sin(\omega t)\,\mathrm{d}t. \tag{2.24}$$

When one applies the inverse Fourier transform operator to the function $Y(\omega)$, the odd extension $y(t)$ is recovered. On the positive real line is the function $x(t)$. This can be succinctly stated as

$$x(t) \equiv F_\mathrm{s}^{-1}[X_\mathrm{s}(\omega)] = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty X_\mathrm{s}(\omega)\sin(\omega t)\,\mathrm{d}\omega. \tag{2.25}$$

Equations (2.24) and (2.25) define an FST pair. Some obvious properties follow:

(a) *Inversion*:

$$F_\mathrm{s} = F_\mathrm{s}^{-1}. \tag{2.26}$$

This is clear from equation (2.25) meaning that,

$$F_\mathrm{s}\{F_\mathrm{s}[x(t)]\} = x(t), \quad \text{for } t \geq 0.$$

(b) *Linearity*:

$$F_\mathrm{s}[\alpha x(t) + \beta y(t)] = \alpha X_\mathrm{s}(\omega) + \beta Y_\mathrm{s}(\omega), \tag{2.27}$$

where α, β are constants. The fact that FST involves integration makes it obvious that it is a linear operator.

(c) *Scaling in time*:

$$F_\mathrm{s}[x(at)] = a^{-1}X_\mathrm{s}\left(\frac{\omega}{a}\right), \quad \text{for real } a > 0. \tag{2.28}$$

This inverse scaling in the frequency domain is obtained in a fashion similar to that in (2.7).

(d) *Shift in time*:

$$F_s[x(t-\alpha)] = F_s[x(t)]\cos(\alpha\omega) + F_c[x(t)]\sin(\alpha\omega). \tag{2.29}$$

As in (2.8a), the result is obtained using the compound angle expansion. By defining an odd extension x_o of the function $x(t)$ such that

$$x_o(t) = \frac{t}{|t|}x(|t|),$$

it is possible to get

$$F_s[x_o(t+\alpha) + x_o(t-\alpha)] = 2X_s(\omega)\cos(\alpha\omega). \tag{2.30}$$

(e) *Shift in frequency*:

$$\begin{aligned} X_s(\omega-\beta) &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty x(t)\sin[(\omega-\beta)t]\,\mathrm{d}t, \quad \text{for } \beta > 0, \\ &= F_s[x(t)\cos(\beta t)] - F_c[x(t)\sin(\beta t)], \end{aligned} \tag{2.31a}$$

and in a very similar way,

$$X_s(\omega+\beta) = F_s[x(t)\cos(\beta t)] + F_c[x(t)\sin(\beta t)].$$

Combining this with (2.31a), we obtain a shift in frequency result involving FST only,

$$F_s[x(t)\cos(\beta t)] = \frac{1}{2}[X_s(\omega-\beta) + X_s(\omega+\beta)]. \tag{2.31b}$$

(f) *Differentiating in time*:

$$\begin{aligned} F_s\left[\frac{\mathrm{d}}{\mathrm{d}t}x(t)\right] &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty \left[\frac{\mathrm{d}}{\mathrm{d}t}x(t)\right]\sin(\omega t)\,\mathrm{d}t, \\ &= \omega F_c[x(t)]. \end{aligned} \tag{2.32}$$

The result is obtained using integration by parts and the fact that sine functions vanish at the origin. That $x(t)$ vanishes as t tends to infinity is also assumed. As in the case of the FCT, even-order derivatives will transform with FST whereas odd-order derivatives will transform with FCT. Note also that differentiation in the time domain is transformed into a multiplication operation in the frequency domain, retaining the simplification property of the Fourier transform.

(g) *Differentiating in frequency*:
Similar to the differentiation in the time domain, we obtain here a reduction from a differentiation operation to a multiplication operation. For the second-order derivative, we have

$$X_{\mathrm{s}}^{(2)}(\omega) = \frac{\mathrm{d}^2}{\mathrm{d}\omega^2}\left\{\left(\frac{2}{\pi}\right)^{1/2}\int_0^\infty x(t)\sin(\omega t)\,\mathrm{d}t\right\}$$
$$= F_{\mathrm{s}}[(-1)t^2 x(t)]. \tag{2.33}$$

As in the case of FCT, odd derivatives in the frequency domain change the transform to FCT. But, as is obvious, the differentiation-to-multiplication transformation is retained. Results similar to (2.11b) and (2.11c) can easily be derived.

(h) *Asymptotic behavior*:
For $x(t)$ being a piecewise continuous and absolutely integrable over the positive real line, the Riemann–Lebesque theorem guarantees that

$$\lim_{\omega\to\infty} X_{\mathrm{s}}(\omega) = 0. \tag{2.34}$$

(i) *Integration in time*:
As opposed to the case for FCT, integration is considered over $[0, t]$ and it gives the following result,

$$F_{\mathrm{s}}\left[\int_0^t x(\tau)\,\mathrm{d}\tau\right] = \frac{1}{\omega}F_{\mathrm{c}}[x(t)]. \tag{2.35}$$

(j) *Integration in frequency*:

$$\int_\omega^\infty X_{\mathrm{s}}(\beta)\,\mathrm{d}\beta = F_{\mathrm{c}}[x(t)/t]. \tag{2.36}$$

Property (h) has been invoked in arriving at this integration result. Note that the property of changing a more complex operation to a simpler one has been maintained under the FST.

(k) *Convolution in time*:
Again, as in the case for FCT, the convolution property for the FST is a little more complicated than that for the Fourier transform. A similar result to (2.16) can be obtained when the odd extended functions for $x(t)$ and $y(t)$ given by

$$x_{\mathrm{o}}(t) = \frac{t}{|t|}x(|t|) \quad \text{and} \quad y_{\mathrm{o}}(t) = \frac{t}{|t|}y(|t|)$$

are convolved. The FCT of the convolution reduces to the product of the FSTs of the two functions $x(t)$ and $y(t)$,

$$2\pi X_{\mathrm{s}}(\omega)Y_{\mathrm{s}}(\omega) = F_{\mathrm{c}}\left\{\int_0^\infty x(\tau)[y(t+\tau) + y_{\mathrm{o}}(t-\tau)]\,\mathrm{d}\tau\right\}. \tag{2.37}$$

The integral on the right-hand side represents the convolution of the odd extended functions. Property (2.37) is obtained by applying the Fourier transform to the convolution and by using the convolution property. The right-hand side results in an FCT because the integral is an even function and the Fourier transform of an even function is directly related to the FCT of that function on the positive real line. Allowing the convolutions of the odd extension of one with the even extension of the other provides the following result,

$$2\pi X_{\mathrm{s}}(\omega)Y_{\mathrm{c}}(\omega) = F_{\mathrm{s}}\left\{\int_0^\infty x(\tau)[y(|t-\tau|) - y(t+\tau)]\,\mathrm{d}\tau\right\}, \tag{2.38a}$$

or

$$2\pi X_{\mathrm{s}}(\omega)Y_{\mathrm{c}}(\omega) = F_{\mathrm{s}}\left\{\int_0^\infty y(\tau)[x(t+\tau) + x_{\mathrm{o}}(t-\tau)]\,\mathrm{d}\tau\right\}. \tag{2.38b}$$

2.5 Some examples of the FST

(a) *The unit rectangular pulse*:
$x(t) = U(t) - U(t-1)$, where $U(t)$ is the Heaviside unit step function.
The FST is given by

$$X_{\mathrm{s}}(\omega) = \left(\frac{2}{\pi}\right)^{1/2}\int_0^1 \sin(\omega t)\,\mathrm{d}t = \left(\frac{2}{\pi}\right)^{1/2}\frac{1-\cos(\omega)}{\omega}. \tag{2.39}$$

(b) *The inverse quadratic function*:

$$x(t) = (t^2 + a^2)^{-1}, \quad a > 0.$$

By using the techniques of contour integration, its FST is obtained as

$$\begin{aligned} X_{\mathrm{s}}(\omega) &= \left(\frac{2}{\pi}\right)^{1/2}\int_0^\infty \frac{1}{(t^2+a^2)}\sin(\omega t)\,\mathrm{d}t, \\ &= \left(\frac{1}{2\pi}\right)^{1/2}\frac{1}{a}[\mathrm{e}^{-a\omega}\overline{Ei}(a\omega) - \mathrm{e}^{a\omega}Ei(-a\omega)]. \end{aligned} \tag{2.40}$$

Here $\overline{Ei}$ and Ei are special functions called the exponential integral functions defined by

$$Ei(\tau) = -\int_{-\tau}^{\infty} \frac{\mathrm{e}^{-t}}{t}\,\mathrm{d}t, \quad |\arg(\tau)| < \pi$$

and

$$\overline{Ei}(\tau) = \frac{1}{2}[Ei(\tau + \mathrm{j}0) + Ei(\tau - \mathrm{j}0)]. \tag{2.41}$$

The result here is somewhat more complicated than that for the FCT.

(c) *The exponential function*:

$$x(t) = \mathrm{e}^{-\alpha t}, \quad \mathrm{Re}(\alpha) > 0.$$

It is easily seen that the FST here is just the Laplace transform of the sine function up to a scale factor,

$$X_{\mathrm{s}}(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^{\infty} \mathrm{e}^{-\alpha t} \sin(\omega t)\,\mathrm{d}t = \left(\frac{2}{\pi}\right)^{1/2} \frac{\omega}{(\alpha^2 + \omega^2)}. \tag{2.42}$$

(d) *The sinc function*:

$$x(t) = \frac{\sin(at)}{t}, \quad a > 0.$$

Its FST is given by

$$\begin{aligned} X_{\mathrm{s}}(\omega) &= \left(\frac{2}{\pi}\right)^{1/2} \int_0^{\infty} \frac{1}{t} \sin(at) \sin(\omega t)\,\mathrm{d}t, \\ &= \left(\frac{1}{2\pi}\right)^{1/2} \ln\left|\frac{\omega + a}{\omega - a}\right|. \end{aligned} \tag{2.43}$$

It is interesting to note here that since $F_{\mathrm{s}}^{-1} = F_{\mathrm{s}}$, the FST of the resulting logarithmic function is immediately seen to be the *sinc* function up to a scale factor.

(e) *The decaying sine function*:

$$x(t) = \mathrm{e}^{-\beta t} \sin(at), \quad a, \mathrm{Re}(\beta) > 0.$$

Its FST may be recognized as the Laplace transform of the function $\sin(at)\sin(\omega t)$ up to a scale factor. Thus,

$$X_s(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty e^{-\beta t} \sin(at)\sin(\omega t)\,dt$$
$$= \left(\frac{1}{2\pi}\right)^{1/2} \beta\left[\frac{1}{\beta^2 + (a-\omega)^2} - \frac{1}{\beta^2 + (a+\omega)^2}\right]. \tag{2.44}$$

(f) *Bessel function of the first kind*:

$$x(t) = J_0(at), \quad a > 0.$$

The result of the FST is very similar to the FCT result, giving,

$$X_s(\omega) = \left(\frac{2}{\pi}\right)^{1/2} \int_0^\infty J_0(at)\sin(\omega t)\,dt$$
$$= 0 \qquad \text{if } 0 < \omega < a,$$
$$= \infty \qquad \text{if } \omega = a,$$
$$= \left(\frac{2}{\pi}\right)^{1/2} (\omega^2 - a^2)^{-1/2} \quad \text{if } \omega > a. \tag{2.45}$$

2.6 The DCTs

As mentioned in the Introduction, DCTs are not simply the discretized versions of the cosine functions. They arise naturally from the discretized solutions of the undamped harmonic oscillator equation together with certain homogeneous boundary conditions. The following discussion follows very closely the presentation made by Strang [1]. Consider first the second-order eigenvalue problem:

$$u'' + \lambda u = 0 \quad \text{on the domain } x \in [0, \pi]. \tag{2.46}$$

The boundary condition of $u'(0) = 0$ at $x = 0$ will generate the $\cos(kx)$ as eigenfunctions with $\lambda = k^2$ as eigenvalues. The additional boundary condition of $u'(\pi) = 0$ at the other end will determine the eigenvalues with $k = 0, \pm 1, \pm 2, \ldots$. Similarly, when the Dirichlet condition is applied at $x = \pi$, i.e., $u(\pi) = 0$, the corresponding values of k are $k = \pm(n + 1/2)$, $n = 0, 1, 2, \ldots$. Note that the eigenfunctions are periodic with a period 2π and symmetric about the boundary points. Suppose (2.46) is to be solved using finite differences. Applying the second central difference to the second-order derivative, on a equispaced grid labeled by the index $l = 0, 1, \ldots, N-1$, we obtain the following difference equation at the grid point l,

$$-u_{l-1} + 2u_l - u_{l+1} = \lambda u_l, \quad \text{for } l \neq 0 \text{ or } N-1. \tag{2.47}$$

The Neumann condition at $x=0$ is translated into $u_{-1}=u_1$. At the other end, the Neumann and Dirichlet conditions are respectively stated as $u_N=u_{N-2}$, and $u_{N-1}=0$. All the boundary conditions are applied at the grid points. When the boundary conditions are applied at the midpoints of the grid, different eigenfunctions and eigenvalues are generated. Fig. 2.1 shows some typical situations of boundary conditions applied either at the grid points or at mid-grid.

The discretized problem with Neumann boundary conditions can now be written in matrix form:

$$\mathbf{A}_1\mathbf{u}_k = \begin{bmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{bmatrix} \mathbf{u}_k = \lambda_k \mathbf{u}_k, \quad k=0,1,\ldots,N-1. \tag{2.48}$$

Using the definition of $u_{lk}=\cos\left(kl\frac{\pi}{N-1}\right)$, as the l-th component of the k-th eigenvector, we can show readily that the k-th eigenvalue is given by

$$\lambda_k = 2-2\cos\left(\frac{k\pi}{N-1}\right), \quad k=0,1,\ldots,N-1. \tag{2.49}$$

Except for a scaling factor of $1/\sqrt{2}$, for $k=0$ and $N-1$, the eigenvectors $\{\vec{u}_k\}$ are the basis functions of the DCT-I. The scaling of the first and last eigenvectors is equivalent to a similarity transformation of the matrix $\mathbf{A}_1$ into a symmetric matrix. Rewriting all of this for the symmetric matrix, we obtain

$$(\mathbf{D}^{-1}\mathbf{A}_1\mathbf{D})(\mathbf{D}^{-1}\mathbf{u}_k) = \begin{pmatrix} 2 & -\sqrt{2} & & & \\ -\sqrt{2} & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -\sqrt{2} \\ & & & -\sqrt{2} & 2 \end{pmatrix} (\mathbf{D}^{-1}\mathbf{u}_k) = \lambda_k(\mathbf{D}^{-1}\mathbf{u}_k) \tag{2.50}$$

where $\mathbf{D}=\text{diag}[\sqrt{2},1,1,\ldots,1,\sqrt{2}]$. The l-th component of the k-th eigenvector is given by

$$(\mathbf{D}^{-1}\mathbf{u}_k)_l = \gamma_l \cos\left(kl\frac{\pi}{N-1}\right) \quad \text{where } \gamma_l = \frac{1}{\sqrt{2}}, \quad \text{for } l=0 \text{ or } N-1,$$
$$= 1, \qquad \text{otherwise.} \tag{2.51}$$

These are precisely the unnormalized basis functions of DCT-I. Since they are the eigenfunctions of a symmetric real matrix, they are necessarily orthogonal. By introducing an additional scaling factor $(N-1)^{-1/2}$ for $k=0$ or $N-1$, and $\sqrt{2}(N-1)^{-1/2}$ otherwise, the basis functions are then normalized.

To introduce the orthonormal transform elements, we first define some scaling factors to be used later:

$$\gamma_l = \frac{1}{\sqrt{2}} \quad \text{for } l=0 \text{ or } N-1; \quad \sigma_l = \frac{1}{\sqrt{2}} \quad \text{for } l=0 \quad \text{and} \quad \varepsilon_l = \frac{1}{\sqrt{2}} \quad \text{for } l=N-1.$$

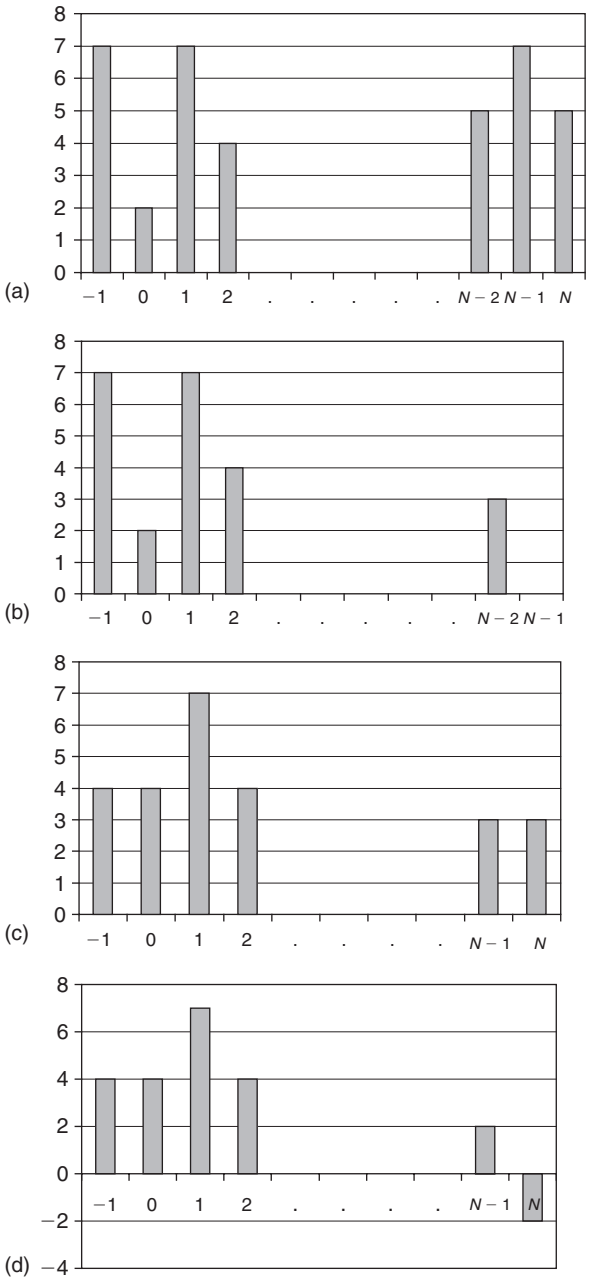


Fig. 2.1. Boundary conditions, $u'(0) = u'(\pi) = 0$ applied at (a) grid points, grid size $= \pi/(N-1)$, $u_{-1} = u_1$ and $u_N = u_{N-2}$; (b) grid points, grid size $= \pi/(N-1)$, $u_{-1} = u_1$ and $u_{N-1} = 0$; (c) mid-grid, grid size $= \pi/N$, $u_{-1} = u_0$ and $u_N = u_{N-1}$ and (d) mid-grid, grid size $= \pi/N$, $u_{-1} = u_0$ and $u_N = -u_{N-1}$.

All these scale factors are unity except as specified. We now introduce a standard notation for the elements of the N-th-order DCT-I transform matrix. Let its lk-th element be denoted by $(\mathbf{C}_N^{\mathrm{I}})_{lk}$ where,

$$(\mathbf{C}_N^{\mathrm{I}})_{lk} = \gamma_k \gamma_l \sqrt{\frac{2}{N-1}} \cos\left(kl \frac{\pi}{N-1}\right), \quad k, l = 0, 1, \ldots, N-1. \tag{2.52}$$

Recall that this result has been obtained by considering the Neumann problem where the boundary conditions are applied at the grid points. The same Neumann problem with the boundary conditions applied mid-grid will generate the elements of the DCT-II. Specifically, the conditions are applied at $l = -1/2$ and $N - 1/2$, giving $u_0 = u_{-1}$ at one end and $u_N = u_{N-1}$ at the other. It should be noted that since the end points are at mid-grid, the grid size is now π/N. The matrix that represents the discretized second-order derivative $\mathbf{A}_2$ is now symmetric and the eigenvalue problem in matrix form is given by

$$\mathbf{A}_2 \mathbf{u}_k = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix} \mathbf{u}_k = \lambda_k \mathbf{u}_k \tag{2.53}$$

Using $\cos[(l + \frac{1}{2})k\pi/N]$ as the l-th component of the k-th eigenvector, it is readily shown that the corresponding eigenvalue is $[2 - 2\cos(k\frac{\pi}{N})]$. These are precisely the elements of the N-th-order DCT-II matrix. They are, when normalized,

$$(\mathbf{C}_N^{\mathrm{II}})_{lk} = \sqrt{\frac{2}{N}} \sigma_k \cos\left[\left(l + \frac{1}{2}\right) \frac{k\pi}{N}\right], \quad k, l = 0, 1, \ldots, N-1. \tag{2.54}$$

It should be pointed out that this derivation of the DCT-II elements does not relate well to the inherent decorrelation power of this well-known transform. The original derivation by Ahmed et al. [5] points directly to the utility of this transform in decorrelating a Markov-1 signal. Our derivation follows the derivation by Strang [1], in preference to the unifying second-order system under examination. Fig. 2.2 shows the basis functions for DCT-II, $N = 8$.

When the Dirichlet condition is used at the far end, we again have two possibilities: both boundary conditions applied at the grid points and both boundary conditions applied mid-grid. The former generates the elements for DCT-III and the latter for DCT-IV. It is interesting to see the matrices corresponding to the second-order derivatives in these cases. We will call them $\mathbf{A}_3$ and $\mathbf{A}_4$, respectively:

$$\mathbf{A}_3 = \begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \quad \text{and} \quad \mathbf{A}_4 = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 3 \end{pmatrix}$$

We note that $\mathbf{A}_3$ is not symmetric. To ensure orthogonality of the eigenvectors, a similarity transformation using the matrix $\mathbf{D} = \mathrm{diag}[\sqrt{2}, 1, \ldots, 1]$ is applied. This results in an extra

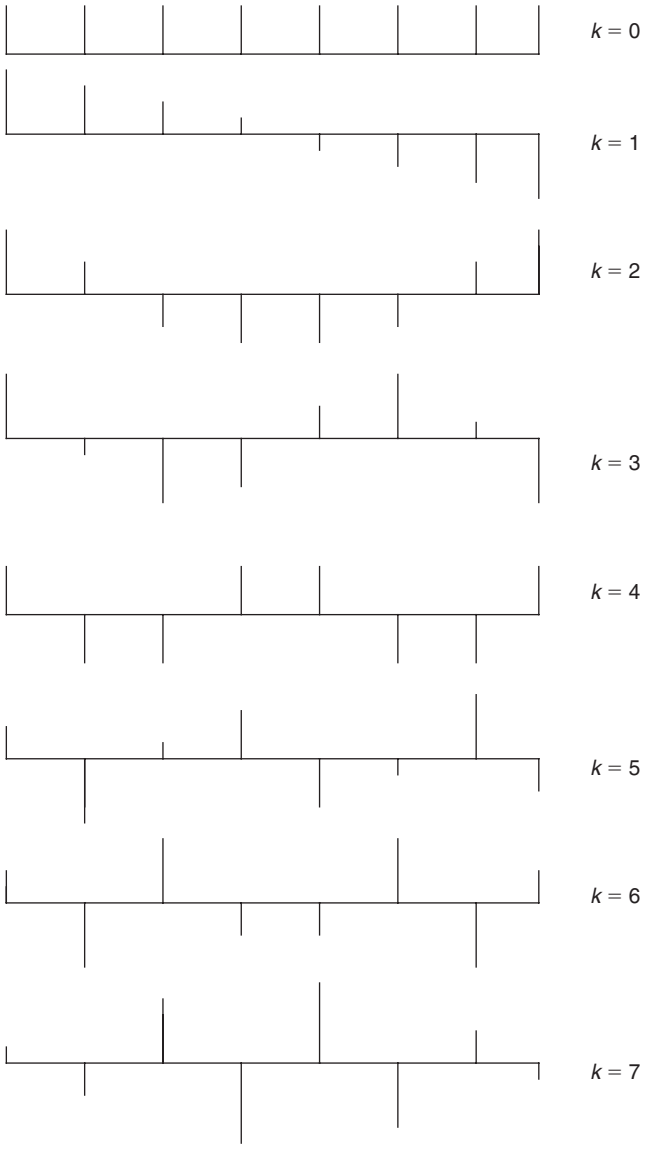


Fig. 2.2. Basis functions for DCT-II, $N = 8$.

scaling factor for the $l = 0$ component of each of the eigenvectors. It is again readily shown that $\cos[l(k+\frac{1}{2})\pi/N]$ and $\cos[(l+\frac{1}{2})(k+\frac{1}{2})\pi/N]$ are respectively the l-th components of the k-th eigenvectors for $\mathbf{A}_3$ and $\mathbf{A}_4$, respectively. We are now ready to write down the elements of the DCT-III and DCT-IV matrices:

$$(\mathbf{C}_N^{\mathrm{III}})_{lk} = \sqrt{\frac{2}{N}}\sigma_l \cos\left[l\left(k+\frac{1}{2}\right)\frac{\pi}{N}\right] \tag{2.55}$$

and

$$(\mathbf{C}_N^{\mathrm{IV}})_{lk} = \sqrt{\frac{2}{N}} \cos\left[\left(l+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{\pi}{N}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.56}$$

These four DCTs are sometimes referred to as the even DCTs. As we see from the above discussion, they are the results of solving the harmonic oscillator system using finite differences. For these DCTs, the boundary conditions at both ends are matched in that they are both applied either at the grid points or at mid-grid. When the boundary conditions are mismatched with one applied at the grid point and the other applied at mid-grid, the so-called odd DCTs result. These are systematically labeled DCT-V to DCT-VIII. The matrices, whose diagonalization will lead to the various DCTs, are listed in the following.

For the Neumann problem with a grid point boundary condition at $x = 0$ and a mid-grid boundary condition at $x = \pi$, we get

$$\mathbf{A}_5 = \begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix}$$

with the boundary conditions applied at the indices $l = 0$ and $j = N - 1/2$, meaning $u_{-1} = u_1$ and $u_N = u_{N-1}$ for the eigenfunctions. As in the case of $\mathbf{A}_1$, to ensure orthogonality of the eigenfunctions, $\mathbf{A}_5$ has to undergo a similarity transformation using the diagonal matrix $\mathbf{D} = \mathrm{diag}[\sqrt{2}, 1, \ldots, 1]$. This results in the $j = 0$ component of each vector being scaled by a factor of $1/\sqrt{2}$. Also, the grid size is now given by $\frac{\pi}{N-\frac{1}{2}}$. The elements of the DCT-V are given by $\cos\left[kl\left(\frac{\pi}{N-\frac{1}{2}}\right)\right]$. These do not make a unitary matrix until we include the normalization factor as well as the scaling factor for the $l = 0$ component. In terms of the usual integer index, the matrix elements for DCT-V are:

$$(\mathbf{C}_N^{\mathrm{V}})_{lk} = \frac{2}{\sqrt{2N-1}} \sigma_l \sigma_k \cos\left[kl\left(\frac{2\pi}{2N-1}\right)\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.57}$$

When the boundary conditions of the above case are reversed in their locations of application, i.e., $x = 0$ applied at $l = -1/2$ and $x = \pi$ at $l = N - 1$, the following matrix is obtained:

$$\mathbf{A}_6 = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{pmatrix}.$$

As can be seen, this matrix again requires a similarity transformation to make it real and symmetric. This is provided by the diagonal matrix $\mathbf{D} = \mathrm{diag}[1, 1, \ldots, \sqrt{2}]$ and hence

the last element of each of the eigenvectors must be scaled by $1/\sqrt{2}$. The elements of the DCT-VI are then given by

$$(\mathbf{C}_N^{\mathrm{VI}})_{lk} = \frac{2}{\sqrt{2N-1}}\sigma_k\varepsilon_l \cos\left[\left(l+\frac{1}{2}\right)k\frac{2\pi}{2N-1}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.58}$$

When the condition at $x=\pi$ is Dirichlet, the same mismatched applications result in the remaining two odd DCTs. Here are the relevant matrices and the elements of the corresponding DCT matrix elements:

$$\mathbf{A}_7 = \begin{pmatrix} 2 & -2 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 3 \end{pmatrix} \quad \text{and} \quad \mathbf{A}_8 = \begin{pmatrix} 1 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

$$\left(\mathbf{C}_N^{\mathrm{VII}}\right)_{lk} = \frac{2}{\sqrt{2N-1}}\varepsilon_k\sigma_l \cos\left[l\left(k+\frac{1}{2}\right)\frac{2\pi}{2N-1}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.59}$$

Similarly, the elements of the DCT-VIII are given by

$$\left(\mathbf{C}_N^{\mathrm{VIII}}\right)_{lk} = \frac{2}{\sqrt{2N+1}} \cos\left[\left(l+\frac{1}{2}\right)\left(k+\frac{1}{2}\right)\frac{2\pi}{2N+1}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.60}$$

We note that the normalization of this vector requires the factor $\sqrt{(2N+1)}$ because the boundary condition at $x=\pi$ is applied at the grid point $l=N$, which is $N+1/2$ units away from $l=-1/2$, the mid-grid at which the $x=0$ boundary condition is applied.

Hence, the eight DCTs are defined by (2.52) and (2.54)–(2.60). It is inherently elegant to see that all come from the diagonalization of matrices having exactly the same tridiagonal interior form. The only distinguishing features are the first and last rows, where the boundary conditions are applied. Only DCT-II has been shown to be optimal in decorrelating a Markov-1 signals. However, it is clear from the common interior structure of these matrices that all DCTs are asymptotically equivalent as N becomes very large, making the effect of the boundary rows insignificant. It should be noted that aside from the scaling factors, DCT-III is the transpose of DCT-II and DCT-VI is the transpose of DCT-VII. Those transform matrices that have a flat vector are specially adapted to analyzing signals with DC components. On the other hand, DCT-IV and DCT-VIII "reach" beyond the boundary and are good candidates as "lapped transforms".

To complete our definitions, we will now examine the derivations for the eight DSTs.

2.7 The DSTs

All that is required to generate the DSTs is simply to replace the Neumann condition at $x=0$ with a Dirichlet condition. In exactly the same way, this condition can be applied

either at a grid point or at mid-grid. These two possibilities with the two possible boundary conditions applied either at grid point or at mid-grid combined to produce the eight DSTs, four even ones and four odd ones, in much the same way the DCTs are derived. The details of the derivation are left as an exercise. We shall denote the matrices to be diagonalized by the DST matrices as $\mathbf{B}_n$. They are listed below:

$$\mathbf{B}_1 = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix}, \quad \mathbf{B}_2 = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 3 \end{pmatrix},$$

$$\mathbf{B}_3 = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{pmatrix}, \quad \mathbf{B}_4 = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix},$$

$$\mathbf{B}_5 = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 3 \end{pmatrix}, \quad \mathbf{B}_6 = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix},$$

$$\mathbf{B}_7 = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{B}_8 = \begin{pmatrix} 3 & -1 & & & \\ -1 & 2 & -1 & & \\ & \bullet & \bullet & \bullet & \\ & & -1 & 2 & -1 \\ & & & -2 & 2 \end{pmatrix}. \tag{2.61}$$

Considering the matrix $\mathbf{B}_1$, we note that it is real and symmetric, rendering an orthogonal set of basis vectors. The matrix has an order of $N-1$, and since the boundary conditions (both are Dirichlet conditions) are applied at the grid points, the size of the grid is given by π/N. Introducing the normalization constant of $\sqrt{2/N}$, the unitary matrix elements of the DST-I can now be written as

$$(\mathbf{S}^{\mathrm{I}}_{N-1})_{lk} = \sqrt{\frac{2}{N}} \sin\Big(lk\frac{\pi}{N}\Big), \quad l,k = 1,2,\ldots,N-1. \tag{2.62}$$

Fig. 2.3 shows the basis functions for DST-I, $N = 8$.

Similar consideration for $\mathbf{B}_2$, which is also real and symmetric, gives the following matrix elements for the DST-II:

$$(\mathbf{S}^{\mathrm{II}}_{N})_{lk} = \sqrt{\frac{2}{N}}\varepsilon_k \sin\left[\frac{(l+\frac{1}{2})(k+1)\pi}{N}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.63}$$

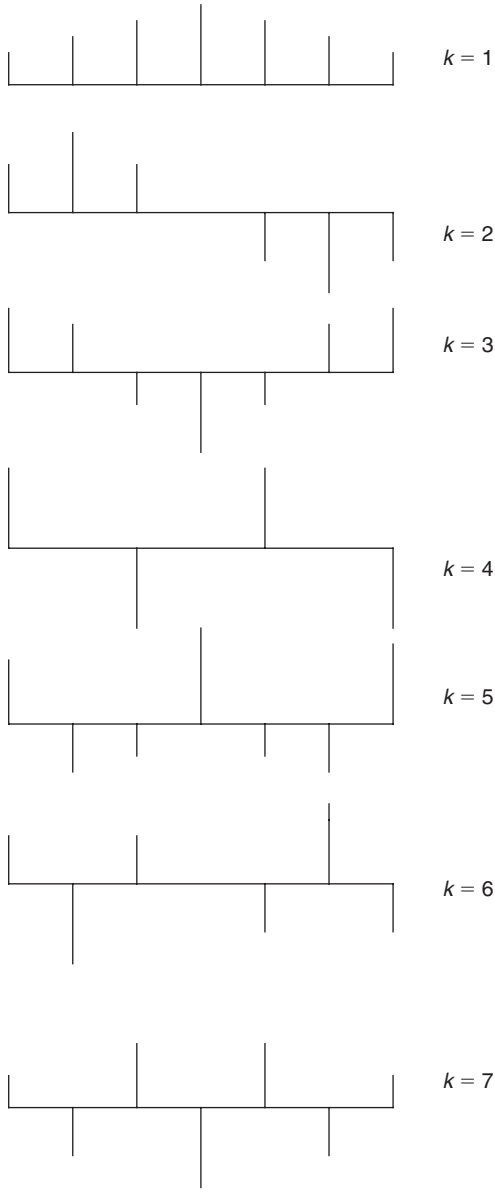


Fig. 2.3. Basis functions for DST-I, $N = 8$.

For $\mathbf{B}_3$, a similarity transformation is required to produce a real and symmetric matrix for orthogonal basis vectors. The transformation matrix is $\mathbf{D} = \text{diag}[1, 1, \ldots, \sqrt{2}]$. The matrix elements for the DST-III are then given by

$$(\mathbf{S}_N^{\text{III}})_{lk} = \sqrt{\frac{2}{N}}\varepsilon_l \sin\left[\frac{(l+1)(k+\frac{1}{2})\pi}{N}\right], \quad l, k = 0, 1, \ldots, N-1. \tag{2.64}$$

For $\mathbf{B}_4$ the matrix is symmetric and real, and hence the eigenvectors are orthogonal, giving the elements for the DST-IV as

$$(\mathbf{S}_N^{\mathrm{IV}})_{lk} = \sqrt{\frac{2}{N}} \sin\left[\frac{(l+\frac{1}{2})(k+\frac{1}{2})\pi}{N}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.65}$$

These are the so-called "even" DSTs. The "odd" DSTs are obtained with similar considerations applied to matrices $\mathbf{B}_5$ to $\mathbf{B}_8$. These are listed in what follows without more detailed discussion:

$$(\mathbf{S}_{N-1}^{\mathrm{V}})_{lk} = \frac{2}{\sqrt{2N-1}} \sin\left[\frac{2lk\pi}{2N-1}\right], \quad l,k = 1,2,\ldots,N-1, \tag{2.66}$$

$$(\mathbf{S}_{N-1}^{\mathrm{VI}})_{lk} = \frac{2}{\sqrt{2N-1}} \sin\left[\frac{2(l+\frac{1}{2})(k+1)\pi}{2N-1}\right], \quad l,k = 0,1,\ldots,N-2, \tag{2.67}$$

$$(\mathbf{S}_{N-1}^{\mathrm{VII}})_{lk} = \frac{2}{\sqrt{2N-1}} \sin\left[\frac{2(l+1)(k+\frac{1}{2})\pi}{2N-1}\right], \quad l,k = 0,1,\ldots,N-2, \tag{2.68}$$

and

$$(\mathbf{S}_N^{\mathrm{VIII}})_{lk} = \frac{2}{\sqrt{2N-1}} \varepsilon_l \varepsilon_k \sin\left[\frac{2(l+\frac{1}{2})(k+\frac{1}{2})\pi}{2N-1}\right], \quad l,k = 0,1,\ldots,N-1. \tag{2.69}$$

Note: Sanchez et al. [16, 17] based on Martucci's results [11–13] have shown that for each version of the DCT and DST there exist a circulant matrix representing symmetric convolution which can be decomposed into the sum of a symmetric Toeplitz matrix and Hankel or close to Hankel matrix scaled by some constant factors. Each DCT and DST can then be generated from the eigenvectors of the decomposed matrix, and the decomposed matrix is thus diagonalized by the appropriate DCT and DST.

2.8 Properties of the DCTs and DSTs

To facilitate applications of DCT and DST in signal processing and other areas, basic properties of these transforms are important. Some properties are intrinsic from the matrix eigenvalue problems which give rise to these transforms, as the derivations in Sections 2.6 and 2.7 demonstrate. Some are similar in nature to those of FCT and FST, while others, such as the convolution properties, require quite a distinctive approach (see Martucci [11–13]). In what follows, some basic properties are discussed for some of the DCTs and DSTs. Similar results can readily be obtained for the remaining DCTs and DSTs:

(a) *The unitarity property*:
The derivations in Sections 2.6 and 2.7 make it quite clear that the DCT and DST (referred collectively sometimes as the discrete trigonometric transforms or DTTs)

transform matrices as defined are unitary. The columns of these transform matrices are eigenvectors of symmetric real matrices. The columns are thus orthogonal. Proper scaling factors as introduced have rendered the column vectors unitary.

As an alternative point of view, the orthonormality of the DCT-I vectors is demonstrated explicitly here.

Recall that the orthonormality of a real N-vector, $\mathbf{u}_k = [u_{0k}, u_{1k}, \ldots, u_{N-1,k}]^T$ is given by the equation

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \delta_{km}, \tag{2.70}$$

where the inner product on the left-hand side is defined by

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \sum_{n=0}^{N-1} u_{nk} u_{nm}, \tag{2.71}$$

and δ_{km} is the Kronecker delta which is zero if k and m are not the same and is unity when they are the same.

Let $\mathbf{u}_k$ be a DCT-I vector as defined in (2.52). We can then write the inner product between two such vectors as

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \sum_{n=0}^{N-1} \left(\frac{2}{N-1} \right) \gamma_k \gamma_n \gamma_n \gamma_m \cos\left(\frac{nk\pi}{N-1} \right) \cos\left(\frac{nm\pi}{N-1} \right), \tag{2.72}$$

where $\gamma_n = 1/\sqrt{2}$ for $n = 0$ or $N-1$, and is unity otherwise. Separating out the first and last terms in the summation and using a trigonometric identity for the cosine function, we obtain

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \frac{\gamma_k \gamma_m}{N-1} \left(1 + (-1)^{m+k} + \sum_{n=1}^{N-2} \left[\cos \frac{(m+k)n\pi}{N-1} + \cos \frac{(m-k)\pi}{N-1} \right] \right). \tag{2.73}$$

The first two terms inside the brackets can be returned into the summation by properly reindexing the summations. This results in

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \frac{\gamma_k \gamma_m}{N-1} \mathrm{Re} \left(\sum_{n=0}^{N-2} W_{2(N-1)}^{-n(m-k)} + \sum_{n=1}^{N-1} W_{2(N-1)}^{-n(m+k)} \right). \tag{2.74}$$

Here, Re $(\cdot)$ denotes "the real part of" and W_M represents the primitive M-th root of unity, i.e.,

$$W_M = \exp\left[-\mathrm{j} \frac{2\pi}{M} \right] = \cos\left(\frac{2\pi}{M} \right) - \mathrm{j} \sin\left(\frac{2\pi}{M} \right). \tag{2.75}$$

Now we examine the two summations in (2.74) separately. For the first summation we set $p = m - k$ to write

$$\sum_{n=0}^{N-2} (W_{2(N-1)}^{-p})^n = \frac{[1 - W_{2(N-1)}^{-(N-1)p}]}{[1 - W_{2(N-1)}^{-p}]} \times \frac{[1 - W_{2(N-1)}^{-p}]^*}{[1 - W_{2(N-1)}^{-p}]^*}$$
$$= \left[2\left(1 - \cos\frac{p\pi}{N-1}\right)\right]^{-1} (1 - W_{2(N-1)}^{-(N-1)p} - W_{2(N-1)}^{p} + W_{2(N-1)}^{-(N-2)p}). \tag{2.76}$$

We have rationalized the complex quotient in preparation for taking the real part. The second summation can be treated the same way. Letting $q = m + k$, we have

$$\sum_{n=1}^{N-1} (W_{2(N-1)}^{-q})^n = \left[2\left(1 - \cos\frac{q\pi}{N-1}\right)\right]^{-1} (W_{2(N-1)}^{-q} - W_{2(N-1)}^{Nq} - 1 + W_{2(N-1)}^{-(N-1)q}) \tag{2.77}$$

Taking the real part of (2.76) gives

$$\mathrm{Re}\left\{\sum_{n=0}^{N-2} (W_{2(N-1)}^{-p})^n\right\} = \frac{[1 - (-1)^{-p}]\left[1 - \cos\frac{p\pi}{N-1}\right]}{2\left[1 - \cos\frac{p\pi}{N-1}\right]} = \frac{1}{2}[1 - (-1)^p],$$

and the real part of (2.77) similarly gives

$$\mathrm{Re}\left\{\sum_{n=1}^{N-1} (W_{2(N-1)}^{-q})^n\right\} = -\frac{1}{2}[1 - (-1)^q].$$

Since p and q differ by $2k$, when p is not zero, i.e., $m \neq k$, the above two expressions cancel and we have

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = 0, \quad \text{for } m \neq k. \tag{2.78}$$

For $m = k (p = 0)$, but $k \neq 0$ or $N - 1$, the result is

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \frac{1}{N-1} \mathrm{Re}\left\{\sum_{n=0}^{N-2} 1 + \sum_{n=1}^{N-1} (W_{2(N-1)}^{-2k})^n\right\} = 1, \tag{2.79}$$

since the second sum is zero for the specified conditions on k. For $k = 0$ or $N - 1$ the scale factors are $1/\sqrt{2}$, and the expression in (2.79) becomes

$$\langle \mathbf{u}_k, \mathbf{u}_m \rangle = \frac{1}{2(N-1)} \mathrm{Re}\left\{\sum_{n=0}^{N-2} 1 + \sum_{n=1}^{N-1} 1\right\} = 1. \tag{2.80}$$

Combining the expressions in (2.78)–(2.80) will complete the proof for unitarity of DCT-I. The tediousness of this proof further demonstrates the elegance of relating these vectors to the eigenvalue problem of the various real symmetric matrices.

Based on their unitarity, the inverses of these transforms can be easily stated as follows

$$\begin{aligned}
&(\mathbf{C}_N^{\mathrm{I}})^{-1} = (\mathbf{C}_N^{\mathrm{I}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{I}}); \qquad (\mathbf{C}_N^{\mathrm{II}})^{-1} = (\mathbf{C}_N^{\mathrm{II}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{III}});\\
&(\mathbf{C}_N^{\mathrm{III}})^{-1} = (\mathbf{C}_N^{\mathrm{III}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{II}}); \qquad (\mathbf{C}_N^{\mathrm{IV}})^{-1} = (\mathbf{C}_N^{\mathrm{IV}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{IV}});\\
&(\mathbf{C}_N^{\mathrm{V}})^{-1} = (\mathbf{C}_N^{\mathrm{V}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{V}}); \qquad (\mathbf{C}_N^{\mathrm{VI}})^{-1} = (\mathbf{C}_N^{\mathrm{VI}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{VII}});\\
&(\mathbf{C}_N^{\mathrm{VII}})^{-1} = (\mathbf{C}_N^{\mathrm{VII}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{VI}}); \quad (\mathbf{C}_N^{\mathrm{VIII}})^{-1} = (\mathbf{C}_N^{\mathrm{VIII}})^{\mathrm{T}} = (\mathbf{C}_N^{\mathrm{VIII}}).
\end{aligned} \tag{2.81}$$

Similar results exist for the eight DSTs. It is of interest to note that of these 8 DCTs only four are involutary, i.e., they are their own inverses. Only these then retain the inversion property of the FCT as stated in (2.5).

(b) *The linearity property*:
All DTTs are obviously linear operators satisfying the same equation as FCT in (2.6).

(c) *The scaling in time property*:
The discrete transforms deal with discrete time samples and the results of the transforms are discrete frequency samples. A scaling in time will result in an inverse scaling in the frequency domain with no impact on the overall transform as in the case of the FCT noted in (2.7).

Based on the time–frequency uncertainty principle, if Δt and Δf are respectively the time and frequency resolutions (or base units), then

$$\Delta t \cdot \Delta f \geq 1/(2\pi). \tag{2.82}$$

Hence when Δt is scaled by a factor a to become $a\Delta t$, the frequency unit Δf must be scaled by the inverse of a to become $(1/a)\Delta f$ so that (2.82) is unchanged. The overall magnitude of the transform remains unchanged.

(d) *The shift in time property*:
The fact that DTTs deal with discrete sample points and finite durations means that the properties which are influenced by the finite end points, such as a shift or a finite convolution will have very different forms [6–8]. It is demonstrated here for the shift in time property for DCT-I.

Let $\mathbf{x} = [x(0), x(1), \ldots, x(N-1)]^T$ and $\mathbf{x}_+ = [x(1), x(2), \ldots, x(N)]^T$ be two sampled data vectors of dimension N. Clearly, $\mathbf{x}_+$ is $\mathbf{x}$ shifted forward by one time unit, or one sample point. While it may be more direct to treat these as completely different data vectors for the purposes of transform analysis, it is nevertheless interesting and useful to see how their transforms are related as compared to the case for FCT in (2.8a).

Referring to (2.52), we can write down the elements of the DCT-I vectors as

$$\begin{aligned}
X^{C_1}(m) &= \sqrt{\frac{2}{N-1}}\gamma_m \sum_{n=0}^{N-1} \gamma_n \cos\left(\frac{mn\pi}{N-1}\right) x(n),\\
X_+^{C_1}(m) &= \sqrt{\frac{2}{N-1}}\gamma_m \sum_{n=0}^{N-1} \gamma_n \cos\left(\frac{mn\pi}{N-1}\right) x(n+1), \quad m = 0, 1, \ldots, N-1.
\end{aligned} \tag{2.83}$$

In matrix-vector form, the transformed vectors are

$$\mathbf{X}^{C_1} = (\mathbf{C}_N^{\mathrm{I}})\mathbf{x} \quad \text{and} \quad \mathbf{X}_+^{C_1} = (\mathbf{C}_N^{\mathrm{I}})\mathbf{x}_+. \tag{2.84}$$

The shift in time property seeks to find some relations between the elements of the two transformed vectors in (2.84). Except for the finiteness of the sampled vectors, the derivation parallels that for FCT. Replacing the index n in the first equation in (2.83) by $(n+1)-1$ and using the compound angle formula for the cosine function, one obtains

$$\begin{aligned} X^{C_1}(m) = \sqrt{\frac{2}{N-1}} \Bigg\{ & \gamma_m \cos\left(\frac{m\pi}{N-1}\right) \sum_{n=0}^{N-1} \gamma_n \cos\left(\frac{m(n+1)\pi}{N-1}\right) x(n+1) \\ & + \gamma_m \sin\left(\frac{m\pi}{N-1}\right) \sum_{n=0}^{N-1} \gamma_n \sin\left(\frac{m(n+1)\pi}{N-1}\right) x(n+1) \Bigg\}. \end{aligned} \tag{2.85a}$$

If we represent the first sum in (2.85a) by C_1 and the second sum by S_1 the expression simplifies to

$$X^{C_1}(m) = \sqrt{\frac{2}{N-1}}\gamma_m \left\{ \cos\left(\frac{m\pi}{N-1}\right) C_1 + \sin\left(\frac{m\pi}{N-1}\right) S_1 \right\}. \tag{2.85b}$$

Were C_1 and S_1, a DCT and a DST, respectively, the similarity with (2.8a) would be complete. Unfortunately, there are some extra terms in both C_1 and S_1 making the relationship a little more complex. We shall examine C_1 first.

$$\begin{aligned} C_1 &= \sum_{n=0}^{N-1} \gamma_n \cos\left(\frac{m(n+1)\pi}{N-1}\right) x(n+1) \\ &= \frac{1}{\sqrt{2}} \cos\left(\frac{m\pi}{N-1}\right) x(1) + \sum_{n=2}^{N-1} \cos\left(\frac{mn\pi}{N-1}\right) x(n) \\ &\quad + \frac{1}{\sqrt{2}}(-1)^m \cos\left(\frac{m\pi}{N-1}\right) x(N). \end{aligned} \tag{2.86a}$$

Noting that the second term here most resembles a DCT-I, we add and subtract the necessary terms to create a DCT-I, giving a final expression for C_1,

$$\begin{aligned} C_1 = \sqrt{\frac{N-1}{2}} \frac{1}{\gamma_m} X^{C_1}(m) - \frac{x(0)}{\sqrt{2}} + \left(\frac{1}{\sqrt{2}} - 1\right) \cos\left(\frac{m\pi}{N-1}\right) x(1) \\ + \left(1 - \frac{1}{\sqrt{2}}\right)(-1)^m x(N-1) + \frac{(-1)^m}{\sqrt{2}} \cos\left(\frac{m\pi}{N-1}\right) x(N). \end{aligned} \tag{2.86b}$$

Similar considerations applied to the term S_1 in constructing a DST-I for the vector yields

$$S_1 = \sqrt{\frac{N-1}{2}} X^{S_1}(m) + \left(\frac{1}{\sqrt{2}} - 1\right) \sin\left(\frac{m\pi}{N-1}\right) x(1) + \frac{(-1)^m}{\sqrt{2}} \sin\left(\frac{m\pi}{N-1}\right) x(N). \tag{2.86c}$$

Putting C_1 and S_1 together in (2.85b) finally produces the shift in time property for DCT-I as

$$X_+^{C_1}(m) = \cos\left(\frac{m\pi}{N-1}\right) X^{C_1}(m) + \gamma_m \sin\left(\frac{m\pi}{N-1}\right) X^{S_1}(m) + \sqrt{\frac{2}{N-1}} \gamma_m \left\{ -\cos\left(\frac{m\pi}{N-1}\right) \left[\frac{x(0)}{\sqrt{2}} - (-1)^m \left(1 - \frac{1}{\sqrt{2}}\right) x(N-1)\right] + \frac{(-1)^m}{\sqrt{2}} x(N) \left[\sin\left(\frac{m\pi}{N-1}\right) + \cos\left(\frac{m\pi}{N-1}\right)\right]\right\}. \tag{2.87}$$

The fact that such a relationship requires a DST as well as other terms involving the end samples renders such a property less attractive. The direct computation of the transform of the shifted sampled vector would, in most cases, be more direct and less complicated. However, under some situations, such as adaptive filtering, the shift in time property may be a useful one. The shift property is also stated here for DCT-II, since this is one of the most effective DTTs. Other shift properties for the remaining DTTs can be derived in a similar way [6, 7]. For DCT-II, we have

$$X_+^{C_2}(m) = \cos\left(\frac{m\pi}{N}\right) X^{C_2}(m) + \sin\left(\frac{m\pi}{N}\right) X^{S_2}(m-1) + \sqrt{\frac{2}{N}} \sigma_m \cos\left(\frac{m\pi}{2N}\right) [(-1)^m x(N) - x(0)]. \tag{2.88}$$

In the above discussion, we have used $X_+^{C_2}(m)$, $X^{C_2}(m)$ and $X^{S_2}(m)$ to denote the m-th elements respectively of the following transformed vectors

$$\mathbf{X}_+^{C_2} = (\mathbf{C}_N^{\mathrm{II}})\mathbf{X}_+, \quad \mathbf{X}^{C_2} = (\mathbf{C}_N^{\mathrm{II}})\mathbf{X} \quad \text{and} \quad \mathbf{X}^{S_2} = (\mathbf{S}_N^{\mathrm{II}})\mathbf{X}. \tag{2.89}$$

Equation (2.88) expresses the shift property of DCT-II if the data sequence is shifted by one sample point only, i.e., $r = 1$, where r is the shift parameter. This result was extended to handle larger step sizes and the general formulae for shift properties of the DCT-II (and corresponding DST-II) were derived generally for any value of the shift parameter $0 < r < N$ to update transformed vectors reflecting r additional data samples and removing r old data samples from the signal [9, 10]. Thus, the size of the shift can be any value between 1 and $N - 1$, where N is the length of data sequence. These shift properties of the DCT-II and DST-II would be useful in applications where time constraints may not permit the immediate processing of every incoming sample point.

(e) *The difference property*:
While there are no direct analogs for the differentiation and integration operations in the domain of discrete signals, there are some applications such as differential pulse code modulation (DPCM) where the differencing of adjacent samples in a signal is required. In such situations, the transform property on differencing may prove useful.

Consider a sequence consisting of differences of adjacent samples in a signal, say,

$$d(n) = x(n+1) - x(n), \quad n = 0, 1, \ldots, N-1$$

which, in vector form is

$$\mathbf{d} = \mathbf{x}_+ - \mathbf{x}. \tag{2.90}$$

It is immediately clear from (2.90) that the shift properties discussed in (d) can be applied. For example, if DCT-II is to be applied to (2.90), one obtains

$$\mathbf{D}^{C_2} \equiv (\mathbf{C}_N^{\mathrm{II}})\mathbf{d} = \mathbf{X}_+^{C_2} - \mathbf{X}^{C_2}. \tag{2.91}$$

The use of (2.88) here may provide some simplification for the elements of $\mathbf{D}^{C_2}$.

2.9 Convolution properties

One of the most powerful, if not the most powerful properties of the Fourier transform in signal processing is its convolution-multiplication property. Martucci [11–13] and Foltz et al. [14, 15] successfully demonstrated that similar properties exist for the family of DTTs.

Such properties depend on the formalization of the notion of symmetric convolution. In the case of discrete Fourier transform (DFT), the convolution-multiplication property exists for what is defined as a "circular" convolution. The need to develop different convolutions for the family of DTTs was perceived for the purpose of producing simple and useful properties. We devote this section to the formalization of symmetric convolutions and the derivation of the corresponding convolution-multiplication properties, following closely Martucci's discussion [11–13].

To begin, it is important to visualize a finite sampled sequence as a portion of an infinite sequence. Any definition of convolution requires the assumed knowledge of how a finite sequence behaves outside its finite support. A very close analogy can be drawn, and in fact exists between this and the Fourier series theory. A function $f(x)$ with a finite support of $[0, \pi]$ may be represented as a Fourier cosine series, or a Fourier sine series, if it has at most a finite number of jump discontinuities. As a cosine series, $f(x)$ is seen as that portion on $[0, \pi]$ of a function of infinite support that is periodic (P) and symmetric (S). As a sine series, this same function is seen as that portion on $[0, \pi]$ of a function of infinite support that is periodic but antisymmetric (A). Hence, the basis functions and therefore the symmetries that are chosen to represent this function of finite support will impact on any definition of the convolution involving this function.

Sampled functions/sequences introduce yet an additional degree of complexity. While the points of symmetry (POS) for a function on a finite but continuous support are always

the end points, the POS of a sampled sequence can be either a grid point (point at which the sample is taken) or at mid-grid (midway between two grid points). Martucci [11–13] refers to these POS as whole sample (W) and half sample (H), respectively.

For each POS, the sampled sequence can therefore be extended as whole sample symmetric (WS), whole sample antisymmetric (WA), half sample symmetric (HS) or half sample antisymmetric (HA). There are always two POSs for a finite sequence, and therefore there will be 16 possibilities of how a finite sequence can be seen as the finite portion of the extended sequence of infinite support. Martucci [11–13] refers to this extended sequence as the symmetric periodic sequence (SPS). It is of no surprise that these 16 combinations correspond exactly to the 16 different eigenvalue problems used by Strang [1] to generate the eight DCTs and eight DSTs (see Sections 2.6 and 2.7). In fact, using the notation of $\varepsilon = (\text{WSHA})$ to denote an SPS extension of the finite sequence with WS at the left POS and HA at the right POS, for instance, one can construct the following table of equivalence relating the types of extension to the DTTs.

The DTTs from type V to VIII are referred to as odd DTTs since one POS is a whole sample and the other is a half sample, dictating that the number of samples points be odd in one period. The corresponding SPS of the DTTs will be the functions that are involved with the convolution-multiplication property of that particular DTT. Before discussing these convolution properties, it is useful to state them in a generic form. Let $\{x(n)\}$ and $\{y(n)\}$ be two sequences or sampled functions and assume that all the relevant transforms exist. Then a generic convolution-multiplication property can be stated as

$$T_1\{x \otimes y\} = T_2\{x\} \times T_3\{y\}, \tag{2.92}$$

where T_1, T_2 and T_3 are transform operators that may not be of the same type, and $\otimes$ denotes a convolution operation. Equation (2.92) holds true when all the transform operators are Fourier transform operators and the convolution is the linear convolution operation for x and y of infinite support. The equation is equally valid when the supports are finite provided that the convolution is defined as a circular convolution. While such a property is particularly elegant when all the transforms are of the same type, its usefulness is not at all diminished if the transforms are of different types. In signal processing, the reduction of the convolution operation in the time (or sample) domain to the multiplication operation in the transform domain is desirable no matter one or more types of transforms are involved.

For finite sequences, their extension SPSs may be symmetric or antisymmetric. In addition, the extension may be periodic (P) with a period N, such that

$$x(n) = x(n+N), \quad n = 0, 1, \ldots, N-1, \tag{2.93a}$$

or it may be antiperiodic (A) with a period N such that

$$x(n) = -x(n+N). \tag{2.93b}$$

It is noted that the symmetric convolution is defined between sequences of the same type of periodicity, i.e., the convolution of a periodic sequence with an antiperiodic sequence

will vanish. We now define the symmetric convolution between two strictly symmetric sequences of period N as the circular convolution given by

$$x(n) \otimes_{\mathrm{c}} y(n) \equiv \sum_{k=0}^{n} x(k)\, y(n-k) + \sum_{k=n+1}^{N-1} x(k)\, y(n-k+N), \quad n = 0, 1, \ldots, N-1. \tag{2.94}$$

Similarly the skew-circular convolution between two antiperiodic sequences is defined by

$$x(n) \otimes_{\mathrm{s}} y(n) \equiv \sum_{k=0}^{n} x(k)\, y(n-k) - \sum_{k=n+1}^{N-1} x(k)\, y(n-k+N). \tag{2.95}$$

Note: An antiperiodic sequence of period N is strictly periodic with a period of $2N$.

In as much as $\{x(n)\}$ and $\{y(n)\}$, for $n = 0, 1, \ldots, N$ are definitive samples of the extended SPSs, the convolution sequences defined in (2.94) and (2.95) must also be interpreted as definitive samples of some SPSs. Thus $\{x(n) \otimes_{\mathrm{c}} y(n)\}$ in (2.94) is the representative finite sequence of a strictly periodic SPS with a period of N, and $\{x(n) \otimes_{\mathrm{s}} y(n)\}$ in (2.95) is the representative finite sequence of an antiperiodic SPS with a period N. It is important to note that when the index of the sequence takes a value beyond the range of $[0, N-1]$, the actual extension ε must be examined to determine the value of the sample. For example, $x(-1) = -x(1)$ if the extension of the sequence $\{x(n)\}$ is WA at the left POS.

Equation (2.92) holds true for the circular convolution when all the transforms are the DFT. The interpretation involving both circular and skew-circular convolutions as well as the various DTTs can be better understood by the introduction of a generalized DFT (GDFT), where arbitrary shifts in both the time and the frequency domains may be applied. Such a GDFT is denoted by $(\mathbf{G}_{a,b})$ and its lk-th element is defined by

$$(\mathbf{G}_{a,b})_{lk} = \exp\left\{-\mathrm{j}(l+b)(k+a)\frac{2\pi}{N}\right\}, \quad l, k = 0, 1, \ldots, N-1. \tag{2.96}$$

Disregarding the scale factors which will make the transforms unitary, we can see that as a and b are allowed to take on the values of either 0 or ½, the real and imaginary parts of $(\mathbf{G}_{a,b})$ are related directly to the various DTTs. For example, the real part of $(\mathbf{G}_{0,1/2})$ is exactly $(\mathbf{C}^{\mathrm{II}})$, when the extension SPS is given by $\varepsilon = (\mathrm{WSWA})$, and the proper scale factors are introduced to make the DCT-II unitary. In contrast, when the extension SPS is $\varepsilon = (\mathrm{WSHA})$, the real part of $(\mathbf{G}_{0,1/2})$ is $(\mathbf{C}^{\mathrm{VI}})$, with the proper scaling factors.

Let the convolutions defined in (2.94) and (2.95) be written as $u(n)$ and $w(n)$, respectively, so that,

$$\begin{aligned} & u(n) = x(n) \otimes_{\mathrm{c}} y(n), \\ \text{and} \quad & w(n) = x(n) \otimes_{\mathrm{s}} y(n), \qquad n = 0, 1, \ldots, N-1. \end{aligned} \tag{2.97}$$

Then, the generic convolution-multiplication relations using the GDFTs of (2.96) can be summarized as follows:

$$\begin{aligned}
&(\mathbf{G}_{0,0})\{u(n)\} = (\mathbf{G}_{0,0})\{x(n)\} \times (\mathbf{G}_{0,0})\{y(n)\},\\
&(\mathbf{G}_{0,1/2})\{u(n)\} = (\mathbf{G}_{0,1/2})\{x(n)\} \times (\mathbf{G}_{0,0})\{y(n)\}\\
\text{and}\quad &(\mathbf{G}_{0,0})\{u(n-1)\} = (\mathbf{G}_{0,1/2})\{x(n)\} \times (\mathbf{G}_{0,1/2})\{y(n)\}, \quad n = 0, 1, \ldots, N-1
\end{aligned} \tag{2.98}$$

for the circular convolutions. For the skew-circular convolutions, we have

$$\begin{aligned}
&(\mathbf{G}_{1/2,0})\{w(n)\} = (\mathbf{G}_{1/2,0})\{x(n)\} \times (\mathbf{G}_{1/2,0})\{y(n)\},\\
&(\mathbf{G}_{1/2,1/2})\{w(n)\} = (\mathbf{G}_{1/2,1/2})\{x(n)\} \times (\mathbf{G}_{1/2,0})\{y(n)\}\\
\text{and}\quad &(\mathbf{G}_{1/2,0})\{w(n-1)\} = (\mathbf{G}_{1/2,1/2})\{x(n)\} \times (\mathbf{G}_{1/2,1/2})\{y(n)\}, \quad n = 0, 1, \ldots, N-1.
\end{aligned} \tag{2.99}$$

Equations (2.98) and (2.99) can now be used to generate all the convolution-multiplication relations for the DTTs, by suitably matching the symmetry, the nature of the periodicity, as well as the positions of the left and right POS, for the underlying extended SPS of the sequence. It is easy to see that (2.98) and (2.99) are the relevant relations for periodic and antiperiodic sequences, respectively.

To conclude this section, we derive a few of the many convolution-multiplication relations for the DTTs to illustrate the procedure involved.

Consider two sequences $\{x(n)\}$ and $\{y(n)\}$, $n = 0, 1, \ldots, N-1$. Let both sequences be symmetrically extended by $\varepsilon = [\text{WSWS}]$, i.e., both POS are whole sample, at grid points and the sequences are symmetrically extended there. This will make both sequences strictly periodic with a period of N. From Table 2.1, we note that the transform that is appropriate for $\varepsilon = [\text{WSWS}]$ is DCT-I. It is also easily verified that $(\mathbf{C}^{\text{I}})$ is the real part of $(\mathbf{G}_{0,0})$ with the proper scaling factors. From the first equation in (2.98) we get the convolution-multiplication relation for DCT-I:

$$(\mathbf{C}^{\text{I}})\{x(n) \otimes_{\text{c}} y(n)\} = (\mathbf{C}^{\text{I}})\{x(n)\} \times (\mathbf{C}^{\text{I}})\{y(n)\}. \tag{2.100}$$

It is thus apparent that DCT-I retains the very nice structure that DFT has in its convolution property.

For our next example, consider that the two sequences are extended by $\varepsilon = [\text{HSHS}]$. Again, from Table 2.1, the equivalent transform is DCT-II. DCT-II can be generated by the real

Table 2.1. Equivalence of extensions to DTTs.

ε	WSWS	HSHS	WSWA	HSHA	WSHS	HSWS	WSHA	HSWA
DCT	I	II	III	IV	V	VI	VII	VIII
ε	WAWA	HAHA	WAWS	HAHS	WAHA	HAWA	WAHS	HAWS
DST	I	II	III	IV	V	VI	VII	VIII

part of the GDFT of $(\mathbf{G}_{0,1/2})$. The underlying SPS for both sequences is strictly periodic with a period N. Using the last equation in (2.98), the convolution-multiplication relation for DCT-II is obtained as

$$(\mathbf{C}^{\mathrm{I}})\{x(n-1) \otimes_{\mathrm{c}} y(n-1)\} = (\mathbf{C}^{\mathrm{II}})\{x(n)\} \times (\mathbf{C}^{\mathrm{II}})\{y(n)\}. \tag{2.101}$$

For the final example, consider a skew-circular convolution. Let both sequences be extended based on $\varepsilon = [\mathrm{HAHS}]$ so that the corresponding SPSs are antiperiodic with a period N. From Table 2.1 we find that the appropriate transform is DST-IV. Now $(\mathbf{S}^{\mathrm{IV}})$ can be obtained from the imaginary part of the GDFT $(\mathbf{G}_{1/2,1/2})$. In fact, $-\mathrm{j}\,(\mathbf{G}_{1/2,1/2})\{x(n)\}$ is exactly $(\mathbf{S}^{\mathrm{IV}})\{x(n)\}$ up to the required scaling factors, when the sequence is extended in this way. Referring to the last equation in (2.99) we note that the real part of $(\mathbf{G}_{1/2,0})$ produces the elements for DCT-III. Hence the convolution-multiplication relation for DST-IV is given by

$$(\mathbf{C}^{\mathrm{III}})\{x(n-1) \otimes_{\mathrm{s}} y(n-1)\} = (\mathbf{S}^{\mathrm{IV}})\{x(n)\} \times (\mathbf{S}^{\mathrm{IV}})\{y(n)\}. \tag{2.102}$$

2.10 Summary

In this chapter, definitions of the eight DCTs and the eight DSTs are provided together with some of their basic properties. While there are useful relations to the FCT and FST, these discrete transforms have been derived as a result of considering the solution of a discretized harmonic oscillator equation with different combinations of boundary conditions and where they are applied. The basic operational properties are capped by a discussion of the convolution-multiplication properties, which are of great importance in the application of these discrete transforms to different areas of digital signal processing.

Problems and Exercises

1. Derive the property (2.9b).
2. Derive the property (2.14).
3. Verify that (2.26) is true.
4. Verify that (2.28) is true.
5. Derive the property (2.35).
6. For DCT-II, show that $(\mathbf{u}_k)_l = \cos[(l+\frac{1}{2})k\pi/N]$ is the l-th element of the k-th eigenvector in (2.53) with $\lambda_k = 2 - 2\cos(k\pi/N)$ as the corresponding eigenvalue.
7. Based on the information given in (2.55) and (2.56), determine the eigenvalues of the $\mathbf{A}_3$ and $\mathbf{A}_4$, respectively, as given in Section 2.6.
8. Show that (2.69) defines the elements of the k-th eigenvector for $\mathbf{B}_8$ in (2.61) and determine the k-th eigenvalue.

9. Following the method outlined in subsection (a) in Section 2.8, demonstrate the unitarity of one of the DCTs other than DCT-I.

10. In subsection (d) of Section 2.8, the shift properties of DST-II and DST-III for windowed sequences similar to (2.88) were obtained by Sherlock and Kakad [9, 10]. Briefly discuss how their results may apply in data sampling and analysis where a finite length data window is used.

11. Determine the appropriate convolution properties using the second equation in (2.98) for the GDFT and all possible combinations of SPS.

References

[1] G. Strang, "The discrete cosine transform", *SIAM Review*, Vol. 41, No. 1, 1999, pp. 135–147.

[2] D. F. Elliott, K. R. Rao, *Fast Transforms: Algorithms, Analyses and Applications*, Academic Press, New York, NY, 1982.

[3] N. Sneddon, *Use of Integral Transforms*, McGraw-Hill, New York, NY, 1972.

[4] A. D. Poularikas, Editor, *The Transforms and Applications Handbook*, CRC&IEEE Press, Boca Raton, FL, 1996, Chapter 3: Sine and cosine transforms, pp. 227–280.

[5] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", *IEEE Transactions on Computers*, Vol. C-23, 1974, pp. 90–93.

[6] P. Yip and K. R. Rao, "On the shift property of DCTs and DSTs", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-35, No. 3, March 1987, pp. 404–406.

[7] L. N. Wu, "Comments on the shift property of DCTs and DSTs", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-38, No. 1, January 1990, pp. 186–188.

[8] J. Xi and J. F. Chicharo, "Computing running DCTs and DSTs based on their second-order shift properties", *IEEE Transactions on Circuits and Systems – I: Fundamental Theory and Applications*, Vol. 47, No. 5, May 2000, pp. 779–783.

[9] B. G. Sherlock and Y. P. Kakad, "Windowed discrete cosine and sine transforms for shifting data", *Signal Processing*, Vol. 81, No. 7, July 2001, pp. 1465–1478.

[10] B. G. Sherlock and Y. P. Kakad, "Transform domain technique for windowing the DCT and DST", *Journal of the Franklin Institute*, Vol. 339, No. 1, January 2002, pp. 111–120.

[11] S. A. Martucci, "Convolution-multiplication properties of the entire family of discrete sine and cosine transforms", *Proceedings of the 26th Annual Conference on Information Sciences and Systems (CISS'92)*, Princeton, NJ, March 1992, pp. 399–404.

[12] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms", *IEEE Transactions on Signal Processing*, Vol. 42, No. 5, 1994, pp. 1038–1051.

[13] S. A. Martucci, "Digital filtering of images using the discrete sine and cosine transform", *Optical Engineering*, Vol. 35, No. 1, January 1996, pp. 119–127.

[14] T. M. Foltz and B. M. Welsh, "Symmetric convolution of asymmetric multidimensional sequences using discrete trigonometric transforms", *IEEE Transactions on Image Processing*, Vol. 8, No. 5, May 1999, pp. 640–651.

[15] T. M. Foltz, B. M. Welsh and C. D. Holmberg, "Symmetric convolution using unitary transform matrices", *IEEE Transactions on Signal Processing*, Vol. 48, No. 9, September 2000, pp. 2691–2692.

[16] V. Sanchez, P. Garcia, A. M. Penaido, J. C. Segura and A. J. Rubio, "Diagonalizing properties of the discrete cosine transforms", *IEEE Transactions on Signal Processing*, Vol. 43, No. 11, November 1995, pp. 2631–2641.

[17] V. Sanchez, A. M. Penaido, J. C. Segura, P. Garcia and A. J. Rubio, "Generating matrices for the discrete sine transforms", *IEEE Transactions on Signal Processing*, Vol. 44, No. 10, October 1996, pp. 2644–2646.

CHAPTER 3

The Karhunen–Loéve Transform and Optimal Decorrelation

3.1 Introduction

In the previous chapter we have presented in a unified fashion the derivations of various discrete trigonometric transforms (DTTs). These transforms have been shown to diagonalize certain tridiagonal matrices having very similar interior structure. However, these derivations provide little or no reason why some of these transforms are so powerful in the digital signal processing area. The original derivation of discrete cosine transform type II (DCT-II) by Ahmed et al. [1] alludes to the decorrelation power of this transform when applied to Markov-1 type random signals. In fact, Ahmed and Flickner [2] and Clarke [23, 24, 27] provided a derivation of DCT-II and discrete sine transform type I (DST-I) based on the diagonalization of the correlation matrix of a stationary Markov signal of high and low correlation coefficient. The wide utility of DCT-II comes from the fact that many real signals do behave statistically like a stationary Markov-1 signal.

The transform that will exactly diagonalize the correlation matrix of any signal is the Karhunen–Loéve transform (KLT), named after Karhunen [3] and Loéve [4], who independently developed the continuous transform. Hotelling's Principal Component Analysis (PCA) [5] is an equivalent approach taken earlier. In essence, the KLT is a series representation of a given random signal, whose orthogonal basis functions are obtained as eigenvectors of the corresponding autocorrelation matrix. This interpretation is examined in Section 3.2, together with a discussion of how this transform will also minimize the mean square error (MSE) of a truncated representation of the actual signal (see Devijer and Kittler [6]). While there are no closed form solutions for the KLT of a general random signal, there is an analytic solution if the signal is stationary Markov-1. Ray and Driver [7] provided this derivation.

Section 3.3 explores the link between DCTs and KLT, specifically, the derivation of DCTs based on the work of Ahmed et al. [2] and that of Kitajima [8, 21]. In Section 3.4, we reflect on the formal treatment of asymptotic equivalence (see Yemini and Pearl [9], Hamidi and

Pearl [25], Jain [26, 28]) between classes of matrices and their orthogonal representations. The discussion leads to the conclusion that DCT-II is asymptotically equivalent to the KLT for Markov signals of all orders. Quadrature approximations are seen to produce actual discrete unitary transforms as well. A summary is provided in Section 3.5.

3.2 The KLT

Let us begin by examining intuitively the problem of transmitting and reconstructing a pure sinusoidal signal over an ideal medium that produces no distortion or degradation. The accuracy of reconstructing the signal certainly depends on how the signal is transmitted. If it is sampled and each sampled value of the signal is sent over the medium, then Shannon's sampling theorem (see Appendix A) determines how many samples per second would be required for the exact duplication of the signal at the receiving end, and this rate of transmission is dependent on the frequency contents of the signal. Yet, from an information point of view, all that is required to reproduce the desired sinusoid are its magnitude, phase and frequency, plus the fact that it is either a sine or cosine signal. Compared to these defining parameters of the sinusoid, even the number of sample points limited by Shannon's theorem will contain a high degree of redundancy. Another way of saying this is that the transmitted sampled data are highly correlated and contain redundant information. A natural question that arises is whether or not it is possible to take the sample data points and transform them into the defining parameters. For a deterministic signal such as our example of a pure sinusoid, the answer is obvious. For random signals, the process is to examine its correlation matrix and to examine its diagonalization, which will produce uncorrelated components. The transform that enables this is the KLT.

Consider a zero mean signal vector of N points given by

$$\mathbf{x} = (x_0, x_1, \ldots, x_{N-1})^{\mathrm{T}}, \tag{3.1}$$

whose samples are correlated. We seek to transform this vector by a linear transformation $\mathbf{W}^{\mathrm{T}}$ so that the transformed vector will have samples that are not correlated. In other words, we are looking for the vector $\mathbf{y}$ given by

$$\mathbf{y} = \mathbf{W}^{\mathrm{T}}\mathbf{x}, \tag{3.2}$$

so that the correlation matrix based on the vector $\mathbf{y}$ is strictly diagonal, or

$$E[\mathbf{y}\mathbf{y}^{\mathrm{T}}] = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}), \tag{3.3}$$

where $\mathrm{diag}(\cdot)$ indicates a diagonal matrix and $E[\cdot]$ is the expectation operator. Substituting (3.2) into (3.3) relates this back to the original vector $\mathbf{x}$ as follows:

$$E[\mathbf{W}^{\mathrm{T}}\mathbf{x}(\mathbf{W}^{\mathrm{T}}\mathbf{x})^{\mathrm{T}}] = \mathbf{W}^{\mathrm{T}}E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]\mathbf{W} = \mathrm{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}). \tag{3.4}$$

Denoting the correlation matrix $E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$ for the vector $\mathbf{x}$ by $\mathbf{A}$, and letting $\mathbf{w}_n$ be the n-th column in the matrix $\mathbf{W}$, the following eigenvalue problem is obtained:

$$\mathbf{A}\mathbf{w}_n = \lambda_n \mathbf{w}_n, \quad n = 0, 1, \ldots, N-1. \tag{3.5}$$

In (3.5) we have applied the fact that the transformation matrix $\mathbf{W}$ is unitary. This is necessary to preserve the energy contained in the vector $\mathbf{x}$ after it has been transformed.

The process encompassed by equations (3.1)–(3.5) can be interpreted in the following way. Given a random signal vector $\mathbf{x}$, its correlation matrix is first estimated using the definition $E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]$. Solving (3.5), this matrix is diagonalized. The transformation $\mathbf{W}$ thus obtained will take the vector $\mathbf{x}$ into its uncorrelated form $\mathbf{y}$ given by (3.2). Note that because the correlation matrix for $\mathbf{x}$ is real and symmetric, the solution of (3.5) will result in a real unitary transform matrix $\mathbf{W}$ (see Appendix A for detail). The eigenvalues in (3.5) are essentially the variances of the vector $\mathbf{y}$. From an information point of view, the higher the variance the higher is the information content. As such, the magnitude of the eigenvalue is an indication of the importance of the corresponding eigenvector $\mathbf{w}_n$ in the reconstruction of the signal vector $\mathbf{x}$. This forms the general basis for many transform domain signal compression schemes and can be understood much more clearly in the following alternative derivation of the KLT from the point of view of MSE.

Consider the vector $\mathbf{x}$ given in (3.1) as an N-dimensional vector. Suppose this vector space is spanned by a set of orthogonal basis vectors $\{\mathbf{w}_n\}$, where $n = 0, 1, \ldots, N-1$. Then it is possible to express the vector $\mathbf{x}$ as a linear combination of these orthogonal basis vectors (see Appendix A), i.e.,

$$\mathbf{x} = \sum_{n=0}^{N-1} \alpha_n \mathbf{w}_n, \tag{3.6}$$

where the coefficients are given by

$$\alpha_n = \langle \mathbf{x}, \mathbf{w}_n \rangle / \langle \mathbf{w}_n, \mathbf{w}_n \rangle, \quad n = 0, 1, 2, \ldots, N-1. \tag{3.7}$$

Thus the vector can be represented either by the actual samples $\{x_n\}$ or the N numbers in (3.7) provided the basis vectors $\{\mathbf{w}_n\}$ are known. Suppose in (3.7) only the first $D (D < N)$ coefficients are significantly different from zero. Then the vector $\mathbf{x}$ will be well represented by these D coefficients, i.e.,

$$\hat{\mathbf{x}} = \sum_{n=0}^{D-1} \alpha_n \mathbf{w}_n, \tag{3.8}$$

may be considered a good approximation of the vector $\mathbf{x}$. It can be seen immediately that if a metric is defined for the error of representation, or the difference between $\mathbf{x}$ and $\hat{\mathbf{x}}$, then the selection of the orthogonal basis functions that span the N-dimensional vector space becomes one of minimizing the error metric. The error metric of choice here is the MSE defined by

$$\varepsilon = E[(\mathbf{x} - \hat{\mathbf{x}})^2]. \tag{3.9}$$

The problem is to find the set of basis vectors $\{\mathbf{w}_n\}$ that will minimize (3.9). If, in addition, the basis vectors are to be normalized, then the additional condition

$$\langle \mathbf{w}_m, \mathbf{w}_n \rangle = \delta_{mn}, \quad m, n = 0, 1, 2, \ldots, N-1, \tag{3.10}$$

must also be satisfied. Equation (3.9) can be rewritten as

$$\varepsilon = E\left[\sum_{n=D}^{N-1} \alpha_n^2\right] = \sum_{n=D}^{N-1} \mathbf{w}_n^{\mathrm{T}} E[\mathbf{x}\mathbf{x}^{\mathrm{T}}]\mathbf{w}_n = \sum_{n=D}^{N-1} \mathbf{w}_n^{\mathrm{T}} \mathbf{A}\mathbf{w}_n. \tag{3.11}$$

Based on these definitions, the variational equation needed to find the basis vectors is

$$\frac{\delta}{\delta \mathbf{w}_n}\{\varepsilon - \lambda_n \langle \mathbf{w}_n, \mathbf{w}_n \rangle\} = 0, \quad n = 0, 1, 2, \ldots, N-1. \tag{3.12}$$

The parameter λ_n is introduced to include the constraint (3.10). When the variational derivative is taken in (3.12), the following equation is obtained:

$$(\mathbf{A} - \lambda_n \mathbf{I})\mathbf{w}_n = 0, \quad n = 0, 1, 2, \ldots, N-1. \tag{3.13}$$

This equation is exactly the same as (3.5), meaning that the vectors $\{\mathbf{w}_n\}$ are exactly the column vectors for the KLT matrix $\mathbf{W}$. The fact that these vectors will provide a truncated representation for the random vector $\mathbf{x}$ which minimizes the MSE is immediately obvious. The MSE due to truncation of using only the first D vectors in $\mathbf{W}$ is given by

$$\varepsilon = \sum_{n=D}^{N-1} \lambda_n, \tag{3.14}$$

which is a minimum for a given D when the eigenvalues are ranked in decreasing order.

It is not difficult to see that there are generally no closed form solutions for the transformation matrix $\mathbf{W}$ since the correlation matrix $\mathbf{A}$ is obviously signal dependent. In the particular case of a stationary Markov-1 signal, Ray and Driver [7] provided the solution. The elements of the correlation matrix of a Markov-1 signal are characterized by

$$[\mathbf{A}]_{mn} = \rho^{|m-n|}, \quad m, n = 0, 1, \ldots, N-1, \tag{3.15}$$

where ρ is the positive adjacent correlation coefficient with a magnitude less than unity. Davenport and Root [10] discussed this in the continuous domain as a solution to a certain integral equation. In the discrete domain, the m-th component of the n-th eigenvector for the solution of (3.13) is given by

$$\mathbf{w}_n(m) = \sqrt{\frac{2}{N + \lambda_n}} \sin\left\{\mu_n\left[(m+1) - \frac{N+1}{2}\right] + \frac{(n+1)\pi}{2}\right\}. \tag{3.16}$$

Here, the eigenvalues are

$$\lambda_n = \frac{1 - \rho^2}{1 - 2\rho\cos(\mu_n) + \rho^2}, \tag{3.17}$$

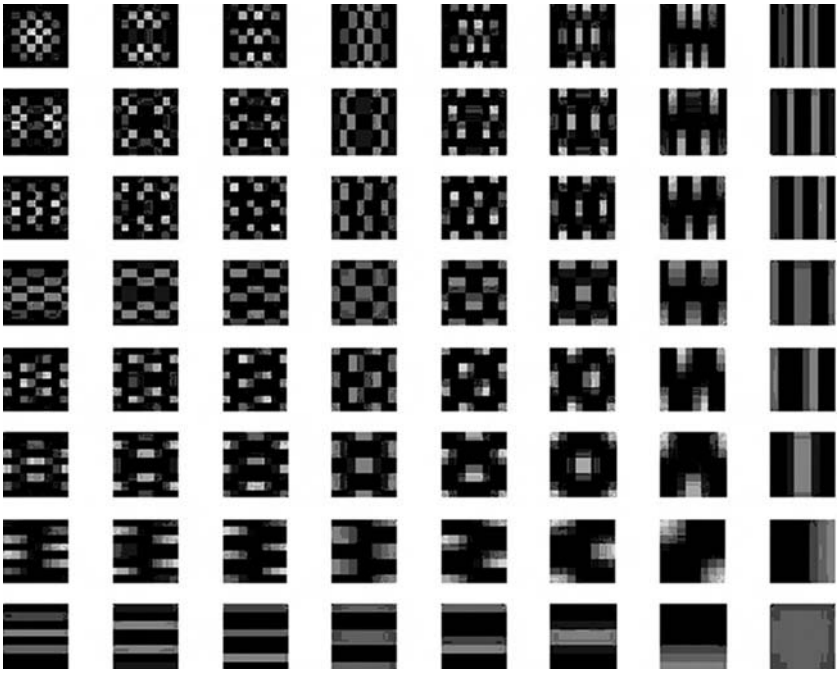


Fig. 3.1. KLT basis functions for the stationary Markov-1 signal with a correlation coefficient ρ of 0.9543 and $N = 8$.

and μ_n's are the real positive roots of the transcendental equation:

$$\tan(N\mu) = -\frac{(1-\rho^2)\sin(\mu)}{(1+\rho^2)\cos(\mu) - 2\rho}. \tag{3.18}$$

Fig. 3.1 shows the KLT basis functions for the stationary Markov-1 signal with a correlation coefficient ρ of 0.9543 and $N = 8$. The intrinsic sinusoidal nature of the functions is quite unmistakable.

Based on the diagonalization of the correlation matrix, the KLT is said to be an optimal transform which:

1. completely decorrelates the signal in the transform domain;
2. minimizes the MSE in bandwidth reduction or data compression;
3. contains the most variance (energy) in the fewest number of transform coefficients;
4. minimizes the total representation entropy of the data sequence.

Despite these ideal properties, its signal dependence and lack of a fast transform algorithm have made it a desirable but impractical tool in signal processing. Since many real signals do possess properties close to those of a stationary Markov-1 signal, a natural question to ask is whether there are predetermined basis vectors that are good approximations to the

KLT. It is in answering this question that attempts had been made to examine the diagonalization of matrices that are asymptotically equivalent to the correlation matrix given by (3.15). The derivations for DCT-I and DCT-II in the next section clearly demonstrate the success of this approach. The KLT for signals of different kinds of statistics cannot be obtained analytically. However, approximation techniques for the basis functions and the eigenvalues have been developed [11]. The KLT also serves well as a benchmark against which other discrete transforms may be judged.

3.3 Asymptotic equivalence of DCT-I and DCT-II to KLT

In the case of a stationary Markov-1 signal, the correlation matrix is a symmetric Toeplitz matrix. It is as we have seen in the last section, one which can be diagonalized in closed form. Symmetric Toeplitz matrices have a great deal of structure and its properties are of great general interest. In fact, a Toeplitz matrix may be treated as asymptotically equivalent (as N, the size of the matrix tends to infinity) to a circulant matrix (see, for example, Davis [12]). It is well known that the eigenvectors of the circulant matrix are the basis vectors of the discrete Fourier transform (DFT). Hence, from this point of view, KLT and DFT are asymptotically equivalent.

A parameter in the stationary Markov-1 signal is its adjacent correlation coefficient ρ. One can also examine the behavior of the correlation matrix and its diagonalization as this coefficient approaches zero or unity. Hence, in speaking about asymptotical equivalence, we will distinguish between this case and the one which is based on the size of the matrix, i.e., N becoming large.

Yet another approach may be taken. Instead of examining the actual correlation matrix $\mathbf{A}$ of the given signal, it is possible to examine an approximation to $\mathbf{A}$ where only a few of the diagonals are retained. If $\mathbf{A}$ is asymptotically equivalent to this approximation, whether in size or in correlation coefficient, then the eigenvalues and eigenvectors of this approximate matrix may be considered to be close to those of the KLT. When this approximation is independent of the signal, the corresponding transform will have the desired property of being predetermined and hence, will be much more practical.

These approaches have been successfully attempted by many researchers (see Refs [2, 8, 9]). From these studies have come not only DCTs and DSTs but also other discrete unitary transforms. We examine the derivations for DCT-I and DCT-II here. Although the derivations lack the elegance of the unified treatment given by Strang [13] presented in the previous chapter, they do relate quite directly to the reasons why these transforms are close to being optimal in decorrelation operations.

3.3.1 DCT-I

To begin, we recall that if a nonsingular matrix $\mathbf{A}$ is diagonalized by a matrix $\mathbf{S}$ in a similarity transformation, then its inverse $\mathbf{A}^{-1}$ is also diagonalized by the same transformation. In other words, if

$$\mathbf{S}^{-1}\mathbf{A}\mathbf{S} = \mathbf{\Lambda} = \text{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}),$$

then

$$\mathbf{S}^{-1}\mathbf{A}^{-1}\mathbf{S} = \mathbf{\Lambda}^{-1} = \text{diag}(\lambda_0^{-1}, \lambda_1^{-1}, \ldots, \lambda_{N-1}^{-1}). \tag{3.19}$$

Another fact that is of importance is that a nonsingular symmetric Toeplitz matrix has a tridiagonal inverse. In particular, for the matrix $\mathbf{A}$ given by (3.15) its inverse is given by

$$\mathbf{A}^{-1} = (1-\rho^2)^{-1} \begin{pmatrix} 1 & -\rho & 0 & \ldots & \ldots & \ldots & \ldots \\ -\rho & 1+\rho^2 & -\rho & \ldots & \ldots & \ldots & \ldots \\ 0 & -\rho & 1+\rho^2 & -\rho & 0 & \ldots & \ldots \\ \ldots & 0 & -\rho & 1+\rho^2 & -\rho & \ldots & \ldots \\ \ldots & \ldots & \ldots & \ldots & -\rho & 1+\rho^2 & -\rho \\ \ldots & \ldots & \ldots & \ldots & \ldots & -\rho & 1 \end{pmatrix}. \tag{3.20}$$

Using the result in (3.19), one now seeks the transformation that will diagonalize (3.20) instead of the matrix in (3.15). One cannot help but notice the similarity of structure between (3.20) and the tridiagonal matrices treated in Section 2.6, where the unified derivation of the DCTs is presented. In this derivation (3.20) is regarded as the sum of two matrices so that

$$\mathbf{A}^{-1} = \mathbf{B} + \mathbf{R},$$

where

$$\mathbf{B} = (1-\rho^2)^{-1} \begin{pmatrix} 1+\rho^2 & -\sqrt{2}\rho & 0 & \ldots & \ldots \\ -\sqrt{2}\rho & 1+\rho^2 & -\rho & \ldots & \ldots \\ 0 & -\rho & 1+\rho^2 & -\rho & \ldots \\ \ldots & \ldots & \ldots & \ldots & -\sqrt{2}\rho \\ \ldots & \ldots & \ldots & -\sqrt{2}\rho & 1+\rho^2 \end{pmatrix},$$

and

$$\mathbf{R} = (1-\rho^2)^{-1} \begin{pmatrix} -\rho^2 & (\sqrt{2}-1)\rho & 0 & \ldots & \ldots \\ (\sqrt{2}-1)\rho & 0 & 0 & \ldots & \ldots \\ \ldots & \ldots & \ldots & 0 & (\sqrt{2}-1)\rho \\ \ldots & \ldots & \ldots & (\sqrt{2}-1)\rho & -\rho^2 \end{pmatrix}. \tag{3.21}$$

We note that $\mathbf{R}$ is close to being a null matrix except at the corners of the main diagonal. Matrix $\mathbf{B}$ is further decomposed into

$$\mathbf{B} = \frac{-2\rho}{1-\rho^2}\mathbf{B}_1 + \frac{1+\rho^2}{1-\rho^2}\mathbf{I}_N. \tag{3.22}$$

Here, $\mathbf{B}_1$ is a very sparse matrix with only two nonzero off diagonals, i.e.,

$$\mathbf{B}_1 = \begin{pmatrix} 0 & \frac{1}{\sqrt{2}} & 0 & \dots & \dots & \dots \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{2} & \dots & \dots & \dots \\ 0 & \frac{1}{2} & 0 & \frac{1}{2} & \dots & \dots \\ \dots & \dots & \frac{1}{2} & 0 & \frac{1}{2} & \dots \\ \dots & \dots & \dots & \dots & \dots & \frac{1}{\sqrt{2}} \\ \dots & \dots & \dots & \dots & \frac{1}{\sqrt{2}} & 0 \end{pmatrix}. \tag{3.23}$$

Based on (3.22) it is seen that the transformation that will diagonalize $\mathbf{B}_1$ will also diagonalize $\mathbf{B}$, since the remaining term in (3.22) is already diagonal.

The eigenvalue problem involving $\mathbf{B}_1$ may be written as

$$\mathbf{B}_1 \mathbf{v}_m = \omega_m \mathbf{v}_m, \quad m = 0, 1, \dots, N-1. \tag{3.24}$$

Note that the eigenvectors $\{\mathbf{v}_m\}$ are also the eigenvectors for $\mathbf{B}$. The solution of (3.24) is based on a three-term recurrence relation for Tchebyshev polynomials of the first kind. These are special functions (see, e.g., Sneddon [14] and Szego [20]) denoted by $T_n(t)$,

$$T_n(t) = \cos(n\theta), \quad \text{where } t = \cos(\theta), \quad n = 0, 1, \dots, N-1. \tag{3.25}$$

The three-term recurrence relation that is relevant here is given by

$$\frac{1}{2} T_{n-1}(t) + \frac{1}{2} T_{n+1}(t) = t T_n(t), \quad \text{for } n = 1, 2, \dots \tag{3.26}$$

While (3.25) and (3.26) are valid for continuous values of t varying between zero and unity, one may consider these equations for discrete values of t, specifically,

$$t_m = \cos\left(\frac{m\pi}{N-1}\right), \quad m = 0, 1, 2, \dots, N-1. \tag{3.27}$$

Applying these discrete values of t to the recurrence relation in (3.26), we obtain

$$T_1(t_m) = t_m T_0(t_m),$$
$$\frac{1}{2} T_{n-1}(t_m) + \frac{1}{2} T_{n+1}(t_m) = t_m T_n(t_m), \quad n = 1, 2, \dots, N-2,$$

and

$$T_{N-2}(t_m) = t_m T_{N-1}(t_m). \tag{3.28}$$

The equations in (3.28) can be easily verified using compound angle formulas for the cosine function.

Consider now a vector whose components are made up of Tchebyshev polynomials so that

$$\tilde{\mathbf{v}}_m = \left(\frac{T_0(t_m)}{\sqrt{2}}, T_1(t_m), T_2(t_m), \ldots, \frac{T_{N-1}(t_m)}{\sqrt{2}} \right)^{\mathrm{T}}, \tag{3.29}$$

which can be normalized by defining

$$\mathbf{v}_m = \frac{\tilde{\mathbf{v}}_m}{|\tilde{\mathbf{v}}_m|}.$$

It is straightforward to see that these are indeed the eigenvectors for the matrix $\mathbf{B}_1$ and the eigenvalues ω_m's are simply the discrete values t_m's. Hence,

$$\mathbf{B}_1 \mathbf{v}_m = t_m \mathbf{v}_m, \quad m = 0, 1, 2, \ldots, N-1. \tag{3.30}$$

Equation (3.30) is just the equations in (3.28) written out in matrix–vector form. By (3.22), the matrix $\mathbf{B}$ is also diagonalized and the equation is

$$\mathbf{B}\mathbf{v}_m = \mu_m \mathbf{v}_m \quad m = 0, 1, 2, \ldots, N-1, \tag{3.31}$$

and the eigenvalues are clearly given by

$$\mu_m = (1-\rho^2)^{-1} \left\{ 1 + \rho^2 - 2\rho \cos\left(\frac{m\pi}{N-1} \right) \right\}. \tag{3.32}$$

As $N \to \infty$, $\mathbf{B}$ will asymptotically approach $\mathbf{A}^{-1}$ as can be shown in the transform domain. Let the matrix $\mathbf{V}$ be

$$\mathbf{V} = [\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{N-1}]^{\mathrm{T}}. \tag{3.33}$$

We note that $\mathbf{V}$ is a unitary matrix and $\mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}_N$. Applying the similarity transformation to the inverse of $\mathbf{A}$, one obtains

$$\mathbf{V}^{\mathrm{T}} \mathbf{A}^{-1} \mathbf{V} = \mathbf{V}^{\mathrm{T}} \mathbf{B} \mathbf{V} + \mathbf{V}^{\mathrm{T}} \mathbf{R} \mathbf{V} = \mathrm{diag}(\mu_0, \mu_1, \ldots, \mu_{N-1}) + \mathbf{V}^{\mathrm{T}} \mathbf{R} \mathbf{V}. \tag{3.34}$$

The nm-th element of the last term in (3.34) can be obtained by direct matrix multiplication as

$$(\mathbf{V}^{\mathrm{T}} \mathbf{R} \mathbf{V})_{nm} = \begin{cases} 0 & \text{for } n+m, \quad \text{odd} \\ 2\gamma_n \gamma_m \rho \dfrac{\left\{ -\rho + (2-\sqrt{2})[\cos \frac{m\pi}{N-1} + \cos \frac{n\pi}{N-1}] \right\}}{(N-1)(1-\rho^2)} & \text{, otherwise,} \end{cases} \tag{3.35}$$

where

$$\gamma_m = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } m = 0 \text{ or } N-1, \\ 1 & \text{otherwise.} \end{cases}$$

Hence, for $\rho \neq 1$ these elements vanish as N tends to infinity and asymptotically the diagonalization of the matrix $\mathbf{A}$ is given by

$$\mathbf{V}^{\mathrm{T}}\mathbf{A}\mathbf{V} \approx \operatorname{diag}(\mu_0^{-1}, \mu_1^{-1}, \ldots, \mu_{N-1}^{-1}) = \operatorname{diag}(\lambda_0, \lambda_1, \ldots, \lambda_{N-1}). \tag{3.36}$$

Therefore, we have an asymptotic solution for (3.5). The eigenvectors, which are the column vectors of the transformation matrix $\mathbf{V}$ are given in terms of (3.29). With proper normalization, the m-th element of the n-th vector is defined by

$$\mathbf{v}_n(m) = \gamma_n \gamma_m \sqrt{\frac{2}{N-1}} \cos\left(\frac{nm\pi}{N-1}\right), \quad n, m = 0, 1, \ldots, N-1. \tag{3.37}$$

Equation (3.37) is identical to equation (2.52) in Chapter 2, which defines the elements of DCT-I. It is noteworthy to point out that as N tends to infinity and ρ tends to one at the same time (3.35) becomes indeterminate, and may not be ignored. The physical interpretation is that for random signals of very high correlation ($\rho \to 1$), DCT-I, which depends on (3.35) being vanishingly small, loses its decorrelation power, since it will no longer be the transformation that diagonalizes the correlation matrix of the Markov-1 signal.

It is interesting to note that the eigenvalues in (3.36) and those for the KLT in (3.17) are identical in form. In fact, if $\mu_n = n\pi/(N-1)$ is used in (3.17) then the basis functions in (3.37) are obtained (see Problem 8). The implication here is that instead of solving (3.18) for the exact μ_n, we have simply approximated it by $n\pi/(N-1)$. However, such a derivation completely masks the asymptotic behavior of the transform.

It should also be pointed out that given a Markov-1 process, the eigenvectors as given by the KLT are predetermined. However, its complex form has made any attempt in deriving fast algorithms quite futile, while the DCT-I and others that are asymptotically equivalent to it (either as ρ tends to one or as N tends to infinity) are amenable to fast algorithms, making real-time implementations feasible. This, in the final analysis is the only justification for the two levels of approximation involved, first that the signal statistics is Markov-1 and secondly the replacement of the basis functions in (3.17) by other simpler ones.

3.3.2 DCT-II

In the derivation for DCT-I, we examine the case where N is large. For DCT-II, we return directly to the KLT for the Markov-1 signal. The eigenvalues and eigenvectors are given in equations (3.16)–(3.18). Applying the limit of $\rho \to 1$ to (3.18), the positive real roots of this transcendental equation are then given by

$$\tan(N\mu) = \lim_{\rho \to 1} \frac{(1-\rho^2)\sin\mu}{(1+\rho^2)\cos\mu - 2\rho} = 0. \tag{3.38}$$

The N real positive roots of (3.38) are then

$$\mu_n = \frac{n\pi}{N}, \quad n = 0, 1, \ldots, N-1. \tag{3.39}$$

Substitution of (3.39) for $n \neq 0$ in (3.17) will yield zero for all the eigenvalues. For $n = 0$ we recall that the trace of a matrix is preserved in a similarity transformation, so that

$$\lim_{\rho \to 1} tr(\mathbf{A}) = \lim_{\rho \to 1} \sum_{n=0}^{N-1} (\mathbf{A})_{nn} = \lim_{\rho \to 1} \sum_{n=0}^{N-1} \lambda_n = \lambda_0. \tag{3.40}$$

Since the diagonal elements of $\mathbf{A}$ are all one's for a Markov-1 signal, λ_0 is immediately found to be N. Returning to the eigenvectors we then obtain

$$\mathbf{w}_0(m) = \frac{1}{\sqrt{N}},$$

and

$$\mathbf{w}_n(m) = \sqrt{\frac{2}{N}} \sin\left\{n(m + \tfrac{1}{2})\frac{\pi}{N} + \frac{\pi}{2}\right\} = \sqrt{\frac{2}{N}} \cos\left\{n(m + \tfrac{1}{2})\frac{\pi}{N}\right\}, \quad n \neq 0. \tag{3.41}$$

Introducing the factor σ_n, the expressions in (3.41) can be combined to give finally the elements of the DCT-II matrix

$$\mathbf{w}_n(m) = \sigma_n \sqrt{\frac{2}{N}} \cos\left\{n(m + \tfrac{1}{2})\frac{\pi}{N}\right\}, \quad m, n = 0, 1, \ldots, N-1, \tag{3.42}$$

where

$$\sigma_n = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } n = 0, \\ 1 & \text{otherwise.} \end{cases}$$

In this particular case, it is worth noting that DCT-II asymptotically behaves like KLT for Markov-1 signals for high correlation independent of the size N of the signal vector. In the case of a perfectly correlated signal, all the variance or energy of the signal is packed in the first eigenvector, which is just the DC component in this limiting case. As ρ decreases, the variance or energy of the signal is spread over the other eigenvectors of the DCT-II. This particular property of DCT-II explains why it is the premier trigonometric transform in signal representation and data compression. In fact, for a *perfectly correlated* signal when $\rho = 1$, the DCT-II basis functions are the eigenvectors of a singular matrix whose elements are all ones (see Problem 9).

Other DCTs and DSTs can be derived in similar fashions. Compared with the unified treatment of the DTTs in Chapter 2, it is easy to see that each of the tridiagonal matrices treated there is asymptotically equivalent to the inverse of the Markov-1 correlation matrix provided the second term in (3.34) vanishes asymptotically.

In the case of DCT-II one can also ask what happens, asymptotically, when N becomes large for a given value of ρ. The answer to this question would follow very closely the procedure that has been used to derive DCT-I. However, there is much economy of thought and elegance of argument in a more general formulation of equivalent classes of matrices and their orthogonal representations. As an example, Gray [15] and Shanmugam [22] have found that circulant matrices and Toeplitz matrices have similar eigenvalue distributions, asymptotically speaking. Pearl [16] undertook a formalization of the so-called asymptotic equivalence of spectral representations in various unitary transform domains. This work was extended by Yemini and Pearl [9] to include a procedure for generating discrete unitary transforms based on the idea of equivalence among classes of matrices. Such a formulation will not only admit the Markov-1 type correlation matrix, but also a much more general class of correlation matrices. The equivalence between this class and the correlation matrix diagonalized by DCT-II, as N becomes large, demonstrates the decorrelation power of DCT-II for Markov signals of all orders. In addition, the ready extension of such a formulation to the actual generation of discrete unitary transforms is sufficient reason for us to devote some space here to its consideration.

3.4 Asymptotic equivalence and generation of discrete unitary transforms

3.4.1 The Hilbert-Schmidt norms of a matrix

The formal discussion of equivalence between classes of matrices rests on the definition of a norm or metric. The vanishing of such a norm as N tends to infinity is usually taken as the necessary and/or sufficient condition for asymptotic equivalence. Since signal correlation matrices are the primary concern here, we shall consider only real symmetric $N \times N$ matrices. Let $\mathbf{A}_N$ be such a matrix, whose eigenvalues are given by λ_n, $n = 0, 1, 2, \ldots, N-1$. The *weak norm* of $\mathbf{A}_N$ is defined as

$$|\mathbf{A}_N|^2 \equiv N^{-1} \sum_{m,n=0}^{N-1} a_{mn}^2 = N^{-1} \sum_{n=0}^{N-1} \lambda_n^2. \tag{3.43}$$

The *strong norm* is defined as

$$||\mathbf{A}_N||^2 \equiv \sup\{\langle \mathbf{u}, \mathbf{A}_N \mathbf{u} \rangle : \langle \mathbf{u}, \mathbf{u} \rangle = 1\} = \max_n \{\lambda_n^2\}. \tag{3.44}$$

Note that these norms are related to the Frobenius norm and the 2-norm, respectively, as defined in Appendix A. It is not difficult to see that the following holds true, i.e.,

$$||\mathbf{A}_N|| \geq |\mathbf{A}_N|. \tag{3.45}$$

As well, these norms are invariant under a unitary transformation $\mathbf{T}$, or

$$|\mathbf{A}_N| = |\mathbf{T}^H \mathbf{A}_N \mathbf{T}| \quad \text{and} \quad ||\mathbf{A}_N|| = ||\mathbf{T}^H \mathbf{A}_N \mathbf{T}||, \tag{3.46}$$

where the superscript H denotes Hermitian conjugation.

3.4.2 Nets, classes and sections

For ease of discussion in treating sequences of matrices of increasing size N, we define the following terms:

1. *Net*: A sequence of strongly bounded matrices $\{\mathbf{A}_N\}$, $N = 1, 2, \ldots, \infty$, denoted by $\boldsymbol{\alpha}$. $||\mathbf{A}_N|| \leq M$, for a finite M and all N.
2. *Class*: A collection of nets denoted by $\mathbf{A}$ with some common structural property. For example, $\mathbf{A}$ is a diagonal class of matrices if it contains nets, each of which is a sequence of strongly bounded diagonal matrices.
3. *N-section*: The collection of $N \times N$ matrices that belong to the nets in a class, denoted by $\mathbf{A}_N$. For example, if $\mathbf{A}$ is a diagonal class, then $\mathbf{A}_N$ is the N-section of this class.

The motivation behind the definitions of these terms is to generalize the concept of asymptotic equivalence from a matrix-to-matrix level to a net-to-net level and, hopefully also to a class-to-class level. Such a generalization provides the necessary framework for determining the equivalence between, for example, the class of circulant matrices and the class of Toeplitz matrices. So, instead of treating the equivalence of one particular matrix to another as in the derivations earlier, a whole class of autocorrelation matrices with a common structural property may be examined.

Starting with *net equivalence*, we say that the two nets $\boldsymbol{\alpha} = \{\mathbf{A}_N\}_{N=1}^{\infty}$ and $\boldsymbol{\beta} = \{\mathbf{B}_N\}_{N=1}^{\infty}$ are equivalent or

$$\boldsymbol{\alpha} \approx \boldsymbol{\beta} \quad \textit{if} \quad \lim_{N\to\infty} |\mathbf{A}_N - \mathbf{B}_N| = 0. \tag{3.47}$$

Thus, the two nets are said to be asymptotically equivalent if a sequence of weak norms of the difference matrices from the two nets vanishes as N tends to infinity. Equation (3.47) indicates that the net equivalence is reflexive. Asymptotic *class equivalence* is a little more complicated. Instead of directly defining equivalence, we introduce first the notion of *asymptotic covering* of classes of matrices. If $\mathbf{A}$ and $\mathbf{B}$ are matrix classes, then $\mathbf{A}$ is said to asymptotically cover $\mathbf{B}$ if for any net $\boldsymbol{\beta} \in \mathbf{B}$, there exists a net $\boldsymbol{\alpha} \in \mathbf{A}$, such that $\boldsymbol{\alpha} \approx \boldsymbol{\beta}$. The definition is given by

$$\mathbf{A} \subset \mathbf{B}, \quad \text{if for any } \boldsymbol{\beta} \in \mathbf{B}, \text{ there exists } \boldsymbol{\alpha} \in \mathbf{A} \text{ such that } \boldsymbol{\alpha} \approx \boldsymbol{\beta}. \tag{3.48}$$

Here, we have introduced the symbol "$\subset$" to denote the asymptotic covering. Note that (3.48) is not necessarily reflexive. Class equivalence is defined if and only if $\mathbf{A} \subset \mathbf{B}$ and $\mathbf{B} \subset \mathbf{A}$. It has been shown, for example, that the class of circulant matrices is asymptotically equivalent to the class of Toeplitz matrices (see Pearl [16]). Hence in the notations that we have introduced

$$\mathbf{A} \approx \mathbf{B} \text{ if and only if } \mathbf{A} \subset \mathbf{B} \text{ and } \mathbf{B} \subset \mathbf{A}. \tag{3.49}$$

The problem of diagonalization of a given signal, correlation matrix can now be formulated more succinctly. Let $\boldsymbol{\tau}$ be a given net of unitary transforms, and let $\mathbf{S}$ be the class of signal correlation matrices. Denote the transformed signal correlation class of matrices by $\boldsymbol{\tau}^H \mathbf{S} \boldsymbol{\tau}$.

The question of diagonalization now becomes whether or not the diagonal class of matrices $\mathbf{D}$ covers the transformed class $\boldsymbol{\tau}^H\mathbf{S}\boldsymbol{\tau}$ asymptotically. In Section 3.4.1, we noted that the weak norm is invariant under a unitary transformation. Hence we see that

$$\mathbf{D} \subset \boldsymbol{\tau}^H\mathbf{S}\boldsymbol{\tau} \text{ if and only if } \boldsymbol{\tau}\mathbf{D}\boldsymbol{\tau}^H \subset \mathbf{S}. \tag{3.50}$$

As the invariance of the weak norm under a unitary transformation, the asymptotic equivalence of $\mathbf{D}$ with $\mathbf{S}$ is now a question of whether $\boldsymbol{\tau}\mathbf{D}\boldsymbol{\tau}^H$ asymptotically covers the class $\mathbf{S}$. To discuss this in terms of an example, we require some properties of spectral representation which we deal with in the next section.

3.4.3 Spectral representations and asymptotic equivalence

In (3.50), $\boldsymbol{\tau}^H\mathbf{S}\boldsymbol{\tau}$ is said to be the $\boldsymbol{\tau}$-spectral representation of the class $\mathbf{S}$. If the transformation results in a diagonal class, $\mathbf{S}$ is said to be class diagonal in the transform net $\boldsymbol{\tau}$. For example, the class diagonal in the Fourier transform net is the class of circulant matrices (see Ref. [7]). It is possible to examine the equivalence of two classes through their respective orthogonal spectral representations. Suppose that $\mathbf{U}$ is the class diagonal in the unitary transform net $\boldsymbol{\tau}$, and $\mathbf{V}$ is the class diagonal in the unitary transform net $\boldsymbol{\beta}$. The asymptotic equivalence between $\mathbf{U}$ and $\mathbf{V}$ can now be examined through the vanishing of a weak norm of some difference matrix class defined using the spectral representations of $\mathbf{U}$ and $\mathbf{V}$.

In essence, if $\mathbf{U} \in \mathbf{U}_N$ and $\mathbf{V} \in \mathbf{V}_N$ (i.e., two individual matrices belonging to the N-sections of the respective classes), and $\mathbf{T}_N \in \boldsymbol{\tau}$, $\mathbf{B}_N \in \boldsymbol{\beta}$, being unitary matrices belonging to the respective nets, the corresponding orthogonal spectral representations are given by

$$\mathbf{T}_N^H\mathbf{U}\mathbf{T}_N = \mathbf{D}_\tau \quad \text{and} \quad \mathbf{B}_N^H\mathbf{V}\mathbf{B}_N = \mathbf{D}_\beta, \tag{3.51}$$

where $\mathbf{D}_\tau = \text{diag}(\tau_0, \tau_1, \ldots, \tau_{N-1})$ and $\mathbf{D}_\beta = \text{diag}(\beta_0, \beta_1, \ldots, \beta_{N-1})$ are diagonal matrices belonging to the N-sections of the classes $\mathbf{D}_\tau$ and $\mathbf{D}_\beta$ respectively. These latter classes are diagonal classes based on the transform nets of $\boldsymbol{\tau}$ and $\boldsymbol{\beta}$. How close the matrix $\mathbf{U}$ is to the matrix $\mathbf{V}$ may be measured by looking at how closely its β-spectral representation is to $\mathbf{D}_\beta$. Another way of putting this is to first consider the β-spectral representation of $\mathbf{U}$ as

$$\mathbf{B}_N^H\mathbf{U}\mathbf{B}_N = \mathbf{B}_N^H(\mathbf{T}_N\, \mathbf{D}_\tau\, \mathbf{T}_N^H)\mathbf{B}_N, \tag{3.52}$$

where the right-hand side of (3.52) may be regarded as the representation of $\mathbf{U}$ in the space spanned by the eigenvectors of $\mathbf{V}$. Let $\{\mathbf{t}\}$ and $\{\mathbf{b}\}$ be the characteristic vectors making up, respectively, the matrices $\mathbf{T}_N$ and $\mathbf{B}_N$. Then $\{\mathbf{t}\}$ completely characterizes $\mathbf{U}_N$ and $\{\mathbf{b}\}$ completely characterizes $\mathbf{V}_N$. The asymptotic equivalence between $\mathbf{U}$ and $\mathbf{V}$ can be examined through $\{\mathbf{t}\}$ and $\{\mathbf{b}\}$. Suppose we define the projection of $\mathbf{U}$ on to the space spanned by $\{\mathbf{b}\}$ as

$$P_{\mathbf{b}}[\mathbf{U}] = \mathbf{T}_N\mathbf{U}'\mathbf{T}_N^H, \tag{3.53}$$

where $\mathbf{U}'$ is used to denote the matrix obtained using only the diagonal elements of (3.52). How close (3.53) is to the original $\mathbf{U}$, or its τ-spectral representation, will indicate the closeness of $\mathbf{U}$ and $\mathbf{V}$. This argument can be understood easily in the special case when

(3.52) is diagonal. When this is true, **U** and **V** are seen to be equivalent since they will be related by a similarity transformation. Using the weak norm for such a measure

$$|\mathbf{U} - P_{\mathbf{b}}[\mathbf{U}]|^2 = \frac{1}{N}\sum_{n=0}^{N-1}\{(\mathbf{D}_\tau)_{nn}^2 - (\mathbf{U}')_{nn}^2\}. \tag{3.54}$$

The vanishing of this weak norm provides the condition for equivalence.

Let the eigenvalues of **U** be denoted by $\mu_{\tau n}$, so that

$$\mu_{\tau n} = (\mathbf{D}_\tau)_{nn}, \tag{3.55}$$

and define a vector using these eigenvalues, i.e.,

$$\boldsymbol{\mu}_\tau = (\mu_{\tau 0}, \mu_{\tau 1}, \ldots, \mu_{\tau(N-1)})^{\mathrm{T}}. \tag{3.56}$$

Then,

$$\sum_{n=0}^{N-1}(\mathbf{D}_\tau)_{nn}^2 = (\boldsymbol{\mu}_\tau)^{\mathrm{T}}(\boldsymbol{\mu}_\tau), \tag{3.57}$$

and

$$\sum_{n=0}^{N-1}(\mathbf{U}')_{nn}^2 = (\boldsymbol{\mu}_\tau)^{\mathrm{T}}\mathbf{A}^{\mathrm{T}}\mathbf{A}(\boldsymbol{\mu}_\tau), \tag{3.58}$$

where the *mn*-th element of the matrix **A** is given by

$$(\mathbf{A})_{mn} = \langle \mathbf{b}_m, \mathbf{t}_n \rangle. \tag{3.59}$$

The above relation is based on (3.52). Putting these into (3.54) we obtain

$$|\mathbf{U} - P_{\mathbf{b}}[\mathbf{U}]|^2 = \frac{1}{N}(\boldsymbol{\mu}_\tau)^{\mathrm{T}}[\mathbf{I} - \mathbf{A}^{\mathrm{T}}\mathbf{A}](\boldsymbol{\mu}_\tau). \tag{3.60}$$

If as $N \to \infty$, and the quantity in (3.60) vanishes, then the classes of matrices characterized by $\{\mathbf{t}\}$ and $\{\mathbf{b}\}$ are said to be asymptotically equivalent. In other words

$$\lim_{N\to\infty}\frac{1}{N}(\boldsymbol{\mu}_\tau)^{\mathrm{T}}[\mathbf{I} - \mathbf{A}^{\mathrm{T}}\mathbf{A}](\boldsymbol{\mu}_\tau) = 0$$

may be used as the condition for the classes of **U** and **V** to be asymptotically equivalent. When this is true for every bounded $\boldsymbol{\mu}_\tau$, a more compact form of the condition results:

$$\lim_{N\to\infty}\left\{1 - \tfrac{1}{N}tr[\mathbf{A}^{\mathrm{T}}\mathbf{A}]\right\} = 0. \tag{3.61}$$

This condition is sufficient for the two classes of matrices to be asymptotically equivalent.

Since we have left out a great deal of subtle details in the above discussion, it is perhaps useful to reflect on this a little in terms of the actual problem in the class of signal correlation matrices. Suppose $\mathbf{U}$ in the above discussion belongs to the N-section of the class $\mathbf{U}$ of signal autocorrelation matrices. Then the net $\boldsymbol{\tau}$ contains the appropriate KLT matrices $\mathbf{T}_N$, since $\mathbf{U}$ is class diagonal with respect to $\boldsymbol{\tau}$. If one wants to examine another transform net, say $\boldsymbol{\beta}$, in which the matrices $\mathbf{B}_N$ may be more readily available than the KLT matrices (3.52) indicates that the diagonal elements of the β-spectral representation of $\mathbf{U}$ should be considered. These diagonal elements are used to construct the projection $P_{\mathbf{b}}[\mathbf{U}]$ which is class diagonal with respect to $\boldsymbol{\tau}$. Hence, the condition (3.61) is used as a sufficient condition for the τ-spectral representation of $\mathbf{U}$ (i.e., in terms of the KLT matrices $\mathbf{T}_N$) and the β-spectral representation of $\mathbf{U}$ (i.e., in terms of the more readily available matrices $\mathbf{B}_N$) to be asymptotically equivalent. In short (3.61) ensures that the matrix $\mathbf{B}_N$ will diagonalize the signal correlation matrix as N tends to infinity. In (3.60), the weak norm of the difference between the matrix $\mathbf{U}$ and its projection may be used to derive a criterion of performance of the $\mathbf{B}_N$ transform matrices.

3.4.4 Gaussian quadrature and generation of transforms

The introduction of concepts of *nets*, *classes* and *sections* in Section 3.4.2 enables us to consider the class of signal correlation matrices and its orthogonal spectral representations in the limit. This limit of $N \to \infty$, coupled with a finite support (assuming that the signal lasts for a finite duration), naturally evokes the transition from the discrete domain to the continuous domain. For example, the inner product of two N-dimensional vector $\mathbf{u}$ and $\mathbf{v}$ is defined as

$$\langle \mathbf{u}, \mathbf{v} \rangle = \sum_{n=0}^{N-1} u_n v_n. \tag{3.62}$$

If $\mathbf{u}$ and $\mathbf{v}$ are vectors of finite duration Ω and N is allowed to increase indefinitely, it is not difficult to see that (3.62) in fact will approach the following inner product of the continuous functions $u(t)$ and $v(t)$ representing the vectors $\mathbf{u}$ and $\mathbf{v}$, respectively,

$$\langle u, v \rangle = \int_{\Omega} u(t)\, v(t)\, \mathrm{d}t. \tag{3.63}$$

Let $\mathbf{u}_m$ and $\mathbf{v}_n$ be the m-th and n-th column vectors of the matrices $\mathbf{U}$ and $\mathbf{V}$, respectively. Then (3.62) and (3.63) represent the term-wise asymptotic behavior of the infinite matrix product, or

$$(\mathbf{U}^H \mathbf{V})_{mn} \underset{N\to\infty}{\longrightarrow} \int_{\Omega} u_m(t) v_n(t) \mathrm{d}t, \tag{3.64}$$

where we have used $u_m(t)$ and $v_n(t)$ to denote the continuous analogs of the discrete vectors $\mathbf{u}_m$ and $\mathbf{v}_n$ as N increases indefinitely over the finite time domain Ω. More generally, instead of $\mathrm{d}t$, a measure given by $\mathrm{d}n(t)$, where $n(t)$ is a distribution can be used. The above discussion makes the link between the infinite matrix product required in examining asymptotic equivalence and quadrature which is the process of evaluating integrals as appearing on the right-hand side of (3.64).

It is now possible to formulate the asymptotic equivalence of two unitary transform nets: $\boldsymbol{\tau} = \{\mathbf{T}_N\}_{N=1}^{\infty}$ and $\boldsymbol{\beta} = \{\mathbf{B}_N\}_{N=1}^{\infty}$. As an example, the former may represent the net of KLT matrices and the latter may be the net of DCT matrices. Thus, if $\mathbf{U}$ is a class of signal correlation matrices, then $\boldsymbol{\tau}^H\ \mathbf{U}\boldsymbol{\tau}$ is a diagonal class of matrices $\boldsymbol{\Lambda}$. To see how closely the net $\boldsymbol{\beta}$ approximates the net $\boldsymbol{\tau}$, it is therefore important first to examine the β-spectral representation of $\boldsymbol{\Lambda}$, as discussed in the previous section, as N increases indefinitely. Hence, we examine

$$(\mathbf{B}_N^H \Lambda_N \mathbf{B}_N)_{mn} \underset{N\to\infty}{\longrightarrow} \int_{\Omega} b_m(t)\lambda(t)b_n(t)\mathrm{d}n(t), \tag{3.65}$$

which is a generalization of (3.64) with a distribution measure $\mathrm{d}n(t)$ and $\lambda(t)$ is used to denote the continuous limit of the vector of the diagonal elements in $\boldsymbol{\Lambda}_N$ as N increases indefinitely. If the right-hand side of (3.65) is, or covers, the signal correlation matrix class $\mathbf{U}$, then $\mathbf{U}$ is asymptotically class diagonal with respect to the net $\boldsymbol{\beta}$. In other words, the net $\boldsymbol{\beta}$ is asymptotically equivalent to the net $\boldsymbol{\tau}$ for the class of signal correlation matrices $\mathbf{U}$. Another way of putting this is to say that

$$\lim_{N\to\infty} \mathbf{B}_N^H \mathbf{U}\mathbf{B}_N = \boldsymbol{\Lambda}. \tag{3.66}$$

So, the class of signal correlation matrices is asymptotically class diagonal in the transform net of DCT matrices.

Two aspects of (3.65) are of interest. First, when the integral represents an element of a matrix belonging to the class $\mathbf{U}$, it should have the structure common to the members of that class. For signal correlation matrices, this common property is the Toeplitz structure. Secondly, the term-wise convergence can be achieved with a finite N if the discrete left-hand side is interpreted as a quadrature evaluation of the integral. It is well known, for example, that the integral of a polynomial of degree $2N-1$ can be exactly reproduced by a numerical quadrature of order N (see, for example, Krylov [17]). It should also be noted that for a given function $\lambda(t)$, and a support Ω, the measure $\mathrm{d}n(t)$ and the orthonormal functions $b_n(t)$'s can be so chosen that the integral in (3.65) will have the necessary Toeplitz structure. For example, with $b_n(t) = \exp(-\mathrm{j}\omega_n t)/(2\pi)$, $\Omega = [-\pi, \pi]$ and $\mathrm{d}n(t) = \mathrm{d}t$, we have

$$(\mathbf{B}_N^H \boldsymbol{\Lambda}_N \mathbf{B}_N)_{mn} \underset{N\to\infty}{\longrightarrow} \left(\frac{1}{2\pi}\right) \int_{-\pi}^{\pi} \exp\{-\mathrm{j}(\omega_n - \omega_m)t\}\lambda(t)\mathrm{d}t,$$

where the right-hand side is clearly the Fourier transform of $\lambda(t)$, denoted by $\tilde{\lambda}(\omega_n - \omega_m)$. As is clear from the form of the argument, the expression has a Toeplitz structure. We are now able to establish a general procedure to generate transform nets that will asymptotically diagonalize the signal correlation matrices:

1. Choose a set of orthonormal functions (polynomials) with respect to a support Ω, and a measure $\mathrm{d}n(t)$ such that the integral in (3.65) will have the required Toeplitz structure.

2. Discretize the orthonormal functions and choose proper weights for the quadrature evaluation of the integral to insure term-wise convergence in (3.65). This will then produce the transform net which asymptotically diagonalizes the class of signal correlation matrices.

Discrete transforms obtained in such a procedure are called Gauss–Jacobi transforms (see Yemini and Pearl [9]).

It is instructive to examine the details of such a construction that will lead to the DCT-II transform matrix. Let the domain of definition of the polynomials $P_m(t)$ be $\Omega = [-1, 1]$ so that they are orthonormal with respect to a certain measure $\mathrm{d}n(t)$. The orthonormality condition for these polynomials is then stated as

$$\int_{\Omega} P_m(t)P_n(t)\mathrm{d}n(t) = \delta_{mn} \quad \text{for } m, n = 0, 1, \ldots, N-1. \tag{3.67}$$

From the theory of numerical quadrature it is known that there exists a set of positive real weights α_n, $n = 0, 1, 2, \ldots, N-1$, corresponding to the N zeros of the polynomial $P_N(t)$ in Ω, denoted by $\{t_n\}$, such that the quadrature of an arbitrary polynomial $f(t)$ given by

$$Q_N(f) = \sum_{n=0}^{N-1} \alpha_n f(t_n), \tag{3.68}$$

is an exact evaluation of the integral

$$\int_{\Omega} f(t)\mathrm{d}n(t),$$

provided the degree of $f(t)$ is less than $2N-1$. Now, in (3.67) it is clear that the integrand is of degree less than $2N-1$. Hence, its quadrature evaluation based on the weights and the N zeros of the polynomial $P_N(t)$ can be used to replace the integral so that

$$\sum_{i=0}^{N-1} \alpha_i P_m(t_i)P_n(t_i) = \delta_{mn}, \quad m, n = 0, 1, \ldots, N-1, \tag{3.69}$$

represents the orthonormality condition in (3.67). The transform net based on this choice of the polynomial function can now be constructed so that the mn-th element of the $N \times N$ transform matrix $\mathbf{B}_N$ is given by

$$(\mathbf{B}_N)_{mn} = \sqrt{\alpha_m} P_n(t_m), \quad m, n = 0, 1, \ldots, N-1. \tag{3.70}$$

We conclude this section by applying the formalism presented to the specific case leading up to the transform matrix for the DCT-II. Let the domain of definition of the polynomials

be $\Omega = [-1,\ 1]$, and let $\mathrm{d}n(t) = \sqrt{1-t^2}\mathrm{d}t$. Then, the polynomials are chosen to be the Tchebyshev polynomials of the first kind, i.e., $P_n(t) = T_n(t) = \cos(n\theta)$, with $\cos(\theta) = t$. To construct the N-point transform, the zeros of $T_N(t)$ are chosen so that $T_N(t_m) = 0$ with

$$t_m = \cos\left(\frac{(2m+1)\pi}{2N}\right), \quad m = 0, 1, \ldots, N-1. \tag{3.71}$$

(see, for example, Fike [18]). The quadrature weights are obtained by observing that the orthogonality condition for the discrete Tchebyshev polynomials is

$$\sum_{m=0}^{N-1} T_i(t_m)T_k(t_m) = \frac{N}{2}\delta_{ik}, \tag{3.72}$$

where the following definitions for the discrete Tchebyshev polynomials have been used

$$T_0(t_m) = \frac{1}{\sqrt{2}}, \quad \text{and} \quad T_k(t_m) = \cos\left(\frac{(2m+1)k\pi}{2N}\right). \tag{3.73}$$

Taking into account the normalization factor and the definition for the polynomial functions chosen, the mn-th element of the transformation matrix is now given as

$$(\mathbf{B}_N)_{mn} = \sqrt{\frac{2}{N}}T_n(t_m),$$

which is clearly the same as the definition of the DCT-II element given in (3.42). As for the matrix class that is class diagonal in this net of transforms, we return to (3.65), in which we use $\Omega = [-1,\ 1]$, $\mathrm{d}n(t) = (1-t^2)^{-\frac{1}{2}}\mathrm{d}t$ and $b_n(t) = T_n(t)$, so that

$$\int_{\Omega} b_m(t)\lambda(t)b_n(t)\mathrm{d}n(t) = \int_{-1}^{1} T_m(t)T_n(t)\lambda(t)(1-t^2)^{-\frac{1}{2}}\,\mathrm{d}t.$$

Substituting the cosine form of the Tchebyshev polynomials and changing the integrating variable to θ reduces the integral to

$$\int_0^{\pi} \cos(m\theta)\cos(n\theta)\lambda(\cos^{-1}\theta)\,\mathrm{d}\theta, \tag{3.74}$$

which can be separated into two parts, using the compound angle formula for the cosine function, so that (3.74) is written as

$$\frac{1}{2}\left\{\int_0^{\pi} \cos[(m-n)\theta]\lambda(\cos^{-1}\theta)\,\mathrm{d}\theta + \int_0^{\pi} \cos[(m+n)\theta]\lambda(\cos^{-1}\theta)\,\mathrm{d}\theta\right\}. \tag{3.75}$$

The first term in (3.75) is clearly of Toeplitz structure while the second term is in what is called a Hankel form (each crossdiagonal consists of the same element). It can be shown that the weak norm of a Hankel form vanishes as N, the size of the matrix, increases, provided certain smoothness condition[1] is satisfied (see Yemini and Pearl in [9]).

This brief discussion on the generation of the DCT-II transform net using the Tchebyshev polynomials demonstrates that it will diagonalize the Toeplitz class of matrices asymptotically. Since this class includes the correlation matrix of Markov signals of all orders, it also means that the DCT-II is asymptotically equivalent to the KLT for Markov signals of all orders, a fact that has important consequences for the DCT-II.

By varying the choices of Ω, $P_n(t)$ and $\mathrm{d}n(t)$ in the quadrature procedure, other discrete unitary transforms such as discrete sine transforms and discrete Legendre transforms can be generated.

3.5 Summary

In this chapter, we have detailed the relationship between the DCT and the statistically optimal KLT. The asymptotic behavior of the DCT is clearly demonstrated both as the sequence length increases and also as the adjacent correlation coefficient increases. Although derivations are presented in details only for DCT-I and DCT-II, the results are similar for other trigonometric transforms. A natural follow-up to this will be the efficient implementation of these transforms based on algorithms that are developed in later chapters.

Section 3.4 deals with the more general problem of so-called equivalent classes of matrices and their orthonormal representations. The emphasis here is on the asymptotic equivalence of different types of correlation matrices. The outcome of the development leads to a rather general and interesting procedure for generating certain discrete unitary transforms for a given class of signal correlation matrices.

Problems and Exercises

1. Derive the equation (3.13).
2. Derive the expression (3.14).
3. Derive the equations (3.16) and (3.18).

---

[1] Consider the Hankel form:

$$\begin{pmatrix} a_0 & a_1 & a_2 & a_3 & \dots \\ a_1 & a_2 & a_3 & a_4 & \dots \\ a_2 & a_3 & a_4 & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots \end{pmatrix}.$$

Its weak norm is given by $\frac{1}{N}\{a_0^2 + 2a_1^2 + 3a_2^2 + \cdots + Na_{N-1}^2\}$, which vanishes as N tends to infinity if the following condition is satisfied: $\sum_{m=1}^{\infty} ma_{m-1}^2 < \infty$. This is referred to as the smoothness condition.

4. Derive the equation (3.20).
5. Derive the equation (3.28).
6. Derive the result in (3.32).
7. Derive the result in (3.35).
8. Toward the end of the derivation for DCT-I in Section 3.3, it is stated that if $\mu_n = n\pi/(N-1)$ is used in (3.17) the basis functions in (3.37) are obtained. Verify this statement.
9. In Section 3.3, on DCT-II, it is stated that the basis functions for DCT-II are the eigenvectors of a singular matrix whose elements are all ones when $\rho = 1$. Prove this.
10. Show that as ρ tends to zero that the basis functions in (3.16) reduces to one of the discrete sine transforms (see Jain [19]).
11. Show that DCT-IV is asymptotic to KLT as N tends to infinity. Investigate the form of the correlation matrix diagonalized by the DCT-IV matrix.

References

[1] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", *IEEE Transactions on Computers*, Vol. C-23, January 1974, pp. 90–93.

[2] N. Ahmed and M. D. Flickner, "Some considerations of the discrete cosine transform", *16th Asilomar Conference on Circuits, Systems and Computers*, Pacific Grove, CA, November 1982, pp. 295–299.

[3] K. Karhunen, "Ueber Lineare Methoden in der Wahrscheinlich-Keitsrechnung", *Annales Academiae Scientiarum Fennicae*, Ser. A137, 1947.

[4] M. Loéve, "Fonctions Aléatoires de Second Ordre", *Processus Stochastiques et Mouvement Brownien,* P. Lévy, Ed. Hermann, Paris, 1948.

[5] H. Hotelling, "Analysis of a complex of statistical variables into principal components", *Journal of Educational Psychology*, Vol. 24, 1933, pp. 417–441 and 498–520.

[6] P. H. Devijer and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall, Englewood Cliffs, New Jersey, NJ, 1982.

[7] W. D. Ray and R. M. Driver, "Further decomposition of the Karhunen-Loéve series representation of a stationary random process", *IEEE Transactions on Information Theory*, Vol. IT-16, November 1970, pp. 663–668.

[8] H. Kitajima, "A symmetric cosine transform", *IEEE Transactions on Computers*, Vol. C-29, April 1980, pp. 317–323.

[9] Y. Yemini and J. Pearl, "Asymptotic properties of discrete unitary transforms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, October 1979, pp. 366–371.

[10] W. D. Davenport and W. L. Root, *An Introduction to the Theory of Random Signals and Noise*, McGraw-Hill, New York, NY, 1958.

[11] J. B. Burl, "Estimating the basis functions of the Karhunen–Loéve transform", *IEEE Transactions on Acoustics Speech and Signal Processing*, Vol. ASSP-37, 1989, pp. 99–105.

[12] P. Davis, *Circulant Matrices*, Wiley, New York, NY, 1979.

[13] G. Strang, "The discrete cosine transform", *SIAM Review*, Vol. 41, No. 1, 1999, pp. 135–147.

[14] I. N. Sneddon, *Special Functions of Mathematical Physics and Chemistry*, Longman Inc., New York, NY, 1980.

[15] R. M. Gray, "On the asymptotic eigenvalue distribution of Toeplitz matrices", *IEEE Transactions on Information Theory*, Vol. IT-18, November 1972, pp. 725–730.

[16] J. Pearl, "Asymptotic equivalence of spectral representations", *IEEE Transactions on Acoustics Speech, and Signal Processing*, Vol. ASSP-23, December 1975, pp. 547–551.

[17] V. I. Krylov, *Approximate Calculation of Integrals*, MacMillan, New York, NY, 1962.

[18] C. T. Fike, *Computer Evaluation of Mathematical Functions*, Prentice Hall, Englewood Cliffs, New Jersey, 1968.

[19] A. K. Jain, "A fast Karhunen–Loéve transform for a class of random processes", *IEEE Transactions on Communications*, Vol. COM-24, September 1976, pp. 1023–1029.

[20] G. Szego, *Orthogonal Polynomials*, American Mathematical Society, New York, 1959.

[21] H. Kitajima, T. Sato and T. Kurobe, "Comparison of the discrete cosine and Fourier transforms as possible substitutes for the Karhunen–Loéve transform", *The Transactions of the IECE of Japan*, Vol. E60, June 1977, pp. 279–283.

[22] K. Sam Shanmugam, "Comments on discrete cosine transform", *IEEE Transactions on Computers*, Vol. C-24, July 1975, p. 759.

[23] R. J. Clarke, "Relation between the Karhunen–Loéve and cosine transforms", *IEE Proceedings*, Vol. 128, Part F: Communications, Radar and Signal Processing, November 1981, pp. 359–360.

[24] R. J. Clarke, "Relation between the Karhunen–Loéve and sine transforms", *Electronics Letters*, Vol. 20, January 1984, pp. 12–13.

[25] M. Hamidi and J. Pearl, "Comparison of the cosine and Fourier transforms of Markov-1 signals", *IEEE Transactions on Acoustics Speech, and Signal Processing*, Vol. ASSP-24, October 1976, pp. 428–429.

[26] A. K. Jain, "A sinusoidal family of unitary transforms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, October 1979, pp. 356–365.

CHAPTER 4

Fast DCT/DST Algorithms

4.1 Introduction

Discrete cosine transforms (DCTs) and discrete sine transforms (DSTs) are members of the class of sinusoidal unitary transforms [13]. A sinusoidal unitary transform is an invertible linear transform whose kernel is defined by a set of complete, orthogonal/orthonormal discrete cosine and/or sine basis functions. The well-known Karhunen-Loéve transform (KLT) [11, 13], generalized discrete Fourier transform (DFT), generalized discrete Hartley transform (DHT) or equivalently generalized discrete W transform, and various types of the DCT and DST are members of this class of unitary transforms. The complete set of DCTs and DSTs, so-called discrete trigonometric transforms, consists of eight versions of DCT and corresponding eight versions of DST [14, 15, 54]. Each transform is classified as even or odd and of type I, II, III and IV. Since all present applications involve only even DCT and DST, this chapter considers only four types of even DCT and DST. The crucial aspect for the applicability of DCT and DST is the existence of fast algorithms that allow their efficient computation compared to the direct matrix–vector multiplication. The algorithmic history of DCT and DST is parallel to that of the DFT. Over three decades many fast algorithms for the efficient computation of one-/two-dimensional (1-D/2-D) DCT and DST have been developed. In general, they are based on indirect computation (via other discrete orthogonal transforms) or direct computation (recursive sparse matrix factorization of the transform matrix), and they are generally classified as radix-2, split-radix, mixed-radix, odd-length, composite-length and prime-factor algorithms [4]. However, one of the most important requirements is that fast DCT/DST algorithms possess excellent numerical stability.

This chapter discusses the fast rotation-based DCT/DST algorithms. Almost all are direct algorithms defined by the recursive (if exists) sparse matrix factorization of the transform matrix. In Section 4.2, definitions, basic mathematical properties and relations between corresponding DCT and DST which are important in deriving fast algorithms are recalled. Then the explicit forms of orthonormal DCT and DST matrices for $N = 2$, 4 and 8 are presented in Section 4.3. The fast and numerically stable DCT/DST algorithms with regular structure employing only real arithmetic are presented in Section 4.4. In particular, these rotation-based algorithms are very convenient for the construction of integer transforms.

The generalized signal flow graphs corresponding to the sparse matrix decomposition of the transform matrix are also provided. Finally, in Section 4.5 how 2-D DCT/DST fast algorithms can be derived from the corresponding 1-D ones is discussed. The chapter concludes with a summary, problems and exercises, and bibliography.

4.2 Orthogonal/orthonormal DCT/DST matrices: definitions, properties and relations

Before deriving the explicit forms of orthonormal DCT and DST matrices we recall their definitions, basic mathematical properties and relations. N is assumed to be an integer power of 2. A subscript in the matrix notation denotes order of the matrix, while a superscript denotes the type.

Four orthonormal even DCTs (forward and inverse) in matrix form denoted by C^{I}_{N+1}, C^{II}_{N}, C^{III}_{N} and C^{IV}_{N}, are respectively defined as [4, 7, 14, 18]

$$[C^{\mathrm{I}}_{N+1}]_{kn} = \sqrt{\frac{2}{N}}\left[\epsilon_k\,\epsilon_n \cos\frac{\pi nk}{N}\right], \qquad k,n = 0,1,\ldots,N,$$

$$[C^{\mathrm{I}}_{N+1}]^{-1} = [C^{\mathrm{I}}_{N+1}]^{\mathrm{T}} = C^{\mathrm{I}}_{N+1}, \tag{4.1}$$

$$[C^{\mathrm{II}}_{N}]_{kn} = \sqrt{\frac{2}{N}}\left[\epsilon_k \cos\frac{\pi(2n+1)k}{2N}\right], \qquad k,n = 0,1,\ldots,N-1,$$

$$[C^{\mathrm{II}}_{N}]^{-1} = [C^{\mathrm{II}}_{N}]^{\mathrm{T}} = C^{\mathrm{III}}_{N}, \tag{4.2}$$

$$[C^{\mathrm{III}}_{N}]_{kn} = \sqrt{\frac{2}{N}}\left[\epsilon_n \cos\frac{\pi(2k+1)n}{2N}\right], \qquad k,n = 0,1,\ldots,N-1,$$

$$[C^{\mathrm{III}}_{N}]^{-1} = [C^{\mathrm{III}}_{N}]^{\mathrm{T}} = C^{\mathrm{II}}_{N}, \tag{4.3}$$

$$[C^{\mathrm{IV}}_{N}]_{kn} = \sqrt{\frac{2}{N}}\left[\cos\frac{\pi(2n+1)(2k+1)}{4N}\right], \qquad k,n = 0,1,\ldots,N-1,$$

$$[C^{\mathrm{IV}}_{N}]^{-1} = [C^{\mathrm{IV}}_{N}]^{\mathrm{T}} = C^{\mathrm{IV}}_{N}, \tag{4.4}$$

where

$$\epsilon_p = \begin{cases} \frac{1}{\sqrt{2}} & p = 0 \text{ or } p = N, \\ 1 & \text{otherwise,} \end{cases}$$

and the corresponding four orthonormal even DSTs (forward and inverse) in matrix form denoted by S^{I}_{N-1}, S^{II}_{N}, S^{III}_{N} and S^{IV}_{N}, are respectively defined as [4, 7, 14, 18]

$$[S^{\mathrm{I}}_{N-1}]_{kn} = \sqrt{\frac{2}{N}}\left[\sin\frac{\pi(n+1)(k+1)}{N}\right], \qquad k,n = 0,1,\ldots,N-2,$$

$$[S^{\mathrm{I}}_{N-1}]^{-1} = [S^{\mathrm{I}}_{N-1}]^{\mathrm{T}} = S^{\mathrm{I}}_{N-1}, \tag{4.5}$$

$$[S_N^{\mathrm{II}}]_{kn} = \sqrt{\frac{2}{N}}\left[\epsilon_k \sin\frac{\pi(2n+1)(k+1)}{2N}\right], \quad k,n = 0,1,\ldots,N-1,$$
$$[S_N^{\mathrm{II}}]^{-1} = [S_N^{\mathrm{II}}]^{\mathrm{T}} = S_N^{\mathrm{III}}, \tag{4.6}$$
$$[S_N^{\mathrm{III}}]_{kn} = \sqrt{\frac{2}{N}}\left[\epsilon_n \sin\frac{\pi(2k+1)(n+1)}{2N}\right], \quad k,n = 0,1\ldots,N-1,$$
$$[S_N^{\mathrm{III}}]^{-1} = [S_N^{\mathrm{III}}]^{\mathrm{T}} = S_N^{\mathrm{II}}, \tag{4.7}$$
$$[S_N^{\mathrm{IV}}]_{kn} = \sqrt{\frac{2}{N}}\left[\sin\frac{\pi(2n+1)(2k+1)}{4N}\right], \quad k,n = 0,1\ldots,N-1,$$
$$[S_N^{\mathrm{IV}}]^{-1} = [S_N^{\mathrm{IV}}]^{\mathrm{T}} = S_N^{\mathrm{IV}}, \tag{4.8}$$

where

$$\epsilon_q = \begin{cases} \frac{1}{\sqrt{2}} & q = N-1, \\ 1 & \text{otherwise.} \end{cases}$$

The DCT-I matrix given by (4.1) is defined for order $N+1$ [9, 14]. It is a scaled version of the symmetric cosine transform (SCT) [10] for $N = 2^m + 1$. The SCT matrix denoted by $\tilde{C}_N^{\mathrm{I}}$ is of order N and it is defined as

$$[\tilde{C}_N^{\mathrm{I}}]_{kn} = \sqrt{\frac{2}{N-1}}\left[\epsilon_k\,\epsilon_n \cos\frac{\pi nk}{N-1}\right], \quad k,n = 0,1,\ldots,N-1,$$
$$[\tilde{C}_N^{\mathrm{I}}]^{-1} = [\tilde{C}_N^{\mathrm{I}}]^{\mathrm{T}} = \tilde{C}_N^{\mathrm{I}}, \tag{4.9}$$

where

$$\epsilon_p = \begin{cases} \frac{1}{\sqrt{2}} & p = 0 \text{ or } p = N-1, \\ 1 & \text{otherwise.} \end{cases}$$

Similarly, the DST-I given by (4.5) is defined for order $N-1$ [13], and it is a scaled version of the symmetric sine transform (SST) [11] for $N = 2^m - 1$. The SST matrix denoted by $\tilde{S}_N^{\mathrm{I}}$ is of order N and it is defined as

$$[\tilde{S}_N^{\mathrm{I}}]_{kn} = \sqrt{\frac{2}{N+1}}\left[\sin\frac{\pi(n+1)(k+1)}{N+1}\right], \quad k,n = 0,1,\ldots,N-1,$$
$$[\tilde{S}_N^{\mathrm{I}}]^{-1} = [\tilde{S}_N^{\mathrm{I}}]^{\mathrm{T}} = \tilde{S}_N^{\mathrm{I}}. \tag{4.10}$$

The DCT-II (and its inverse, the DCT-III, first reported in [8]) has excellent energy compaction and among the currently known unitary transforms it is the best approximation to the optimal KLT. The DST-II and its inverse, the DST-III, were introduced by Kekre and

Solanki [12]. The DCT-IV and DST-IV introduced by Jain [13] have found applications in the fast implementation of lapped orthogonal transforms and cosine/sine modulated filter banks for efficient transform/subband coding [5].

Jain [13] has shown that the basis vectors of DCTs and DSTs are eigenvectors of the parametrized symmetric tridiagonal Jacobi matrix. However, these DCTs and DSTs are not complete. The complete set of DCTs and DSTs was reported in Ref. [14]. Recently, Strang [15] proved that the column vectors of DCTs and DSTs (even and odd versions) are eigenvectors of a simple symmetric second difference matrix (it is actually a tridiagonal matrix), and therefore they are orthogonal. By varying the boundary conditions the complete set of DCTs and DSTs are obtained [54]. Similarly, in Refs. [16, 17] it has been shown that the set of DCTs and DSTs can be generated by decomposing the matrix as a sum of symmetric Toeplitz matrix, Hankel or near-Hankel matrix scaled by a constant factor. Furthermore, the DCTs and DSTs are intrinsically related to generalized DFT [19], and generalized DHT [20, 21] or equivalently generalized discrete W transform [22] for real-valued data [18, 23].

The DCT and DST matrices given by (4.1)–(4.10) are real-valued and orthonormal. The normalization factors $\sqrt{2/N}$, $\sqrt{2/(N-1)}$ and $\sqrt{2/(N+1)}$ in the forward and inverse transforms can be merged as $2/N$, $2/(N-1)$ and $2/(N+1)$, respectively, and moved either to the forward or inverse transform. If these normalization factors are merged these DCT and DST matrices will only be orthogonal. The inverses of DCT and DST matrices are simply obtained by transposing original matrices (unitarity property). The matrices C_{N+1}^{I}, C_N^{IV}, S_{N-1}^{I}, S_N^{IV} including $\tilde{C}_N^{\mathrm{I}}$ and $\tilde{S}_N^{\mathrm{I}}$ are involutory, i.e., they are self-inverse. The symmetry of the transform matrix indicates that the fast algorithms for the forward and inverse transform computation are identical. The matrices C_N^{II} and C_N^{III} are inverses (transposes) of each other. The same property holds for matrices S_N^{II} and S_N^{III}. It means that fast algorithms for the inverse transform computation are obtained from the algorithms for forward transform computation performed in the opposite direction.

To reduce the set of DCT and DST matrices considered for the efficient implementation we can exploit the relations between DST and their corresponding DCT matrices. Specifically, the matrix S_N^{II} (S_N^{III}) is related to C_N^{II} (C_N^{III}) matrix by [39, 40]

$$S_N^{\mathrm{II}} = J_N C_N^{\mathrm{II}} D_N, \quad S_N^{\mathrm{III}} = D_N C_N^{\mathrm{III}} J_N, \tag{4.11}$$

whereas the matrix S_N^{IV} is related to C_N^{IV} matrix by [40]

$$S_N^{\mathrm{IV}} = J_N C_N^{\mathrm{IV}} D_N, \tag{4.12}$$

where J_N is the cross-indentity (reflection) matrix and D_N is the diagonal odd-sign changing matrix given by $D_N = \mathrm{diag}\{(-1)^k\}, k = 0, 1, \ldots, N-1$. Consequently, the efficient implementations of DST-II and DST-IV can be obtained from those of DCT-II and DCT-IV respectively by appropriate sign changes and reversal of order.

The matrix C_N^{IV} is related to C_N^{II} matrix by [48, 49]

$$C_N^{\mathrm{IV}} = \begin{pmatrix} \frac{1}{2} & 0 & 0 & \cdots & 0 \\ -\frac{1}{2} & 1 & 0 & \cdots & 0 \\ \frac{1}{2} & -1 & 1 & \cdots & 0 \\ -\frac{1}{2} & 1 & -1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -\frac{1}{2} & 1 & -1 & \cdots & 1 \end{pmatrix} C_N^{\mathrm{II}} \begin{pmatrix} 2\cos\frac{\pi}{4N} & & & 0 \\ & 2\cos\frac{3\pi}{4N} & & \\ & & \ddots & \\ 0 & & & 2\cos\frac{(N-1)\pi}{4N} \end{pmatrix}. \tag{4.13}$$

Finally, for the DCT and DST matrices the following relations hold [55]:

$$\begin{aligned} C_{N+1}^{\mathrm{I}} J_{N+1} = D_{N+1} C_{N+1}^{\mathrm{I}}, \quad & S_{N-1}^{\mathrm{I}} J_{N-1} = D_{N-1} S_{N-1}^{\mathrm{I}}, \\ C_N^{\mathrm{II}} J_N = D_N C_N^{\mathrm{II}}, \quad & S_N^{\mathrm{II}} J_N = D_N S_N^{\mathrm{II}}, \\ J_N C_N^{\mathrm{III}} = C_N^{\mathrm{III}} D_N, \quad & J_N S_N^{\mathrm{III}} = S_N^{\mathrm{III}} D_N, \\ (-1)^{N-1} C_N^{\mathrm{IV}} J_N D_N = J_N D_N C_N^{\mathrm{IV}}, \quad & (-1)^{N-1} S_N^{\mathrm{IV}} J_N D_N = J_N D_N S_N^{\mathrm{IV}}. \end{aligned} \tag{4.14}$$

These relations allow us to limit our discussion in subsequent sections to C_{N+1}^{I}, S_{N-1}^{I}, C_N^{II}, C_N^{IV}, $\tilde{C}_N^{\mathrm{I}}$ and $\tilde{S}_N^{\mathrm{I}}$.

4.3 The explicit forms of orthonormal DCT/DST matrices

In this section, we derive the explicit orthonormal forms of DCT/DST matrices for some values of N. We consider only the matrices C_{N+1}^{I}, S_{N-1}^{I}, $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$, C_N^{II} and C_N^{IV}, and we will subsequently derive their explicit orthonormal forms for values of $N = 2$, 4 and 8. The matrices C_N^{III}, S_N^{II}, S_N^{III} and S_N^{IV} can be easily derived exploiting the mathematical properties of DCT and DST matrices and relations between DST and their corresponding DCT matrices. Having derived the explicit forms of the DCT and DST matrices for values of $N = 2$, 4 and 8, the higher-order matrices can be generated from lower-order ones by recursive sparse matrix factorization.

The elements of DCT-I matrix C_{N+1}^{I} are defined by (4.1). For values of $N = 2$, 4 and 8, we have the following explicit forms:

$$C_3^{\mathrm{I}} = \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{pmatrix}, \quad C_5^{\mathrm{I}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \cos\frac{\pi}{4} & 0 & -\cos\frac{\pi}{4} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & 0 & -1 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & -\frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{2} \end{pmatrix},$$

$$C_9^{\mathrm{I}} = \frac{1}{2}\begin{pmatrix}
\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\
\frac{1}{\sqrt{2}} & \cos\frac{\pi}{8} & \cos\frac{\pi}{4} & \sin\frac{\pi}{8} & 0 & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \cos\frac{\pi}{4} & 0 & -\cos\frac{\pi}{4} & -1 & -\cos\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{8} & 0 & \cos\frac{\pi}{8} & \cos\frac{\pi}{4} & -\sin\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & 0 & -1 & 0 & 1 & 0 & -1 & 0 & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & \cos\frac{\pi}{8} & 0 & -\cos\frac{\pi}{8} & \cos\frac{\pi}{4} & \sin\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\cos\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & -1 & \cos\frac{\pi}{4} & 0 & -\cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{4} & -\sin\frac{\pi}{8} & 0 & \sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & \cos\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{2}
\end{pmatrix}.$$

The elements of the DST-I matrix S_{N-1}^{I} are defined by (4.5). For $N = 2$, the matrix $S_1^{\mathrm{I}} = (1)$ is trivial and for $N = 4$ and 8, we have the following forms:

$$S_3^{\mathrm{I}} = \frac{1}{\sqrt{2}}\begin{pmatrix}
\sin\frac{\pi}{4} & 1 & \sin\frac{\pi}{4} \\
1 & 0 & -1 \\
\sin\frac{\pi}{4} & -1 & \sin\frac{\pi}{4}
\end{pmatrix},$$

$$S_7^{\mathrm{I}} = \frac{1}{2}\begin{pmatrix}
\sin\frac{\pi}{8} & \sin\frac{\pi}{4} & \cos\frac{\pi}{8} & 1 & \cos\frac{\pi}{8} & \sin\frac{\pi}{4} & \sin\frac{\pi}{8} \\
\sin\frac{\pi}{4} & 1 & \sin\frac{\pi}{4} & 0 & -\sin\frac{\pi}{4} & -1 & -\sin\frac{\pi}{4} \\
\cos\frac{\pi}{8} & \sin\frac{\pi}{4} & -\sin\frac{\pi}{8} & -1 & -\sin\frac{\pi}{8} & \sin\frac{\pi}{4} & \cos\frac{\pi}{8} \\
1 & 0 & -1 & 0 & 1 & 0 & -1 \\
\cos\frac{\pi}{8} & -\sin\frac{\pi}{4} & -\sin\frac{\pi}{8} & 1 & -\sin\frac{\pi}{8} & -\sin\frac{\pi}{4} & \cos\frac{\pi}{8} \\
\sin\frac{\pi}{4} & -1 & \sin\frac{\pi}{4} & 0 & -\sin\frac{\pi}{4} & 1 & -\sin\frac{\pi}{4} \\
\sin\frac{\pi}{8} & -\sin\frac{\pi}{4} & \cos\frac{\pi}{8} & -1 & \cos\frac{\pi}{8} & -\sin\frac{\pi}{4} & \sin\frac{\pi}{8}
\end{pmatrix}.$$

The elements of the SCT matrix $\tilde{C}_N^{\mathrm{I}}$ are defined by (4.9). For values of $N = 2$, 4 and 8, we have the following forms:

$$\tilde{C}_2^{\mathrm{I}} = \sqrt{2}\begin{pmatrix}
\frac{1}{2} & \frac{1}{2} \\
\frac{1}{2} & -\frac{1}{2}
\end{pmatrix}, \quad \tilde{C}_4^{\mathrm{I}} = \sqrt{\frac{2}{3}}\begin{pmatrix}
\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\
\frac{1}{\sqrt{2}} & \cos\frac{\pi}{3} & -\cos\frac{\pi}{3} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\cos\frac{\pi}{3} & -\cos\frac{\pi}{3} & \frac{1}{\sqrt{2}} \\
\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2}
\end{pmatrix},$$

$$\tilde{C}_8^{\mathrm{I}} = \sqrt{\frac{2}{7}} \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{3\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{2\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{\pi}{7} & -\cos\frac{\pi}{7} & \cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2} \end{pmatrix}.$$

The elements of the SST matrix $\tilde{S}_N^{\mathrm{I}}$ are defined by (4.10). For values of $N = 2$, 4 and 8, we have the following forms:

$$\tilde{S}_2^{\mathrm{I}} = \sqrt{\frac{2}{3}} \begin{pmatrix} \sin\frac{\pi}{3} & \sin\frac{2\pi}{3} \\ \sin\frac{2\pi}{3} & -\sin\frac{\pi}{3} \end{pmatrix}, \quad \tilde{S}_4^{\mathrm{I}} = \sqrt{\frac{2}{5}} \begin{pmatrix} \sin\frac{\pi}{5} & \sin\frac{2\pi}{5} & \sin\frac{2\pi}{5} & \sin\frac{\pi}{5} \\ \sin\frac{2\pi}{5} & \sin\frac{\pi}{5} & -\sin\frac{\pi}{5} & -\sin\frac{2\pi}{5} \\ \sin\frac{2\pi}{5} & -\sin\frac{\pi}{5} & -\sin\frac{\pi}{5} & \sin\frac{2\pi}{5} \\ \sin\frac{\pi}{5} & -\sin\frac{2\pi}{5} & \sin\frac{2\pi}{5} & -\sin\frac{\pi}{5} \end{pmatrix},$$

$$\tilde{S}_8^{\mathrm{I}} = \frac{\sqrt{2}}{3} \begin{pmatrix} \sin\frac{\pi}{9} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{4\pi}{9} & \sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{9} \\ \sin\frac{2\pi}{9} & \sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{\pi}{9} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & -\sin\frac{2\pi}{9} \\ \sin\frac{\pi}{3} & \sin\frac{\pi}{3} & 0 & -\sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & \sin\frac{\pi}{3} \\ \sin\frac{4\pi}{9} & \sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{2\pi}{9} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & -\sin\frac{4\pi}{9} \\ \sin\frac{4\pi}{9} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & \sin\frac{2\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & \sin\frac{4\pi}{9} \\ \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} \\ \sin\frac{2\pi}{9} & -\sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & \sin\frac{2\pi}{9} \\ \sin\frac{\pi}{9} & -\sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & \sin\frac{4\pi}{9} & -\sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & -\sin\frac{\pi}{9} \end{pmatrix}.$$

The elements of the DCT-II matrix C_N^{II} are defined by (4.2). For values of $N = 2$, 4 and 8, we have the following forms:

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad C_4^{\mathrm{II}} = \frac{1}{\sqrt{2}} \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{pmatrix},$$

$$C_8^{\mathrm{II}} = \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & -\sin\frac{\pi}{16} \end{pmatrix}.$$

Finally, the elements of the DCT–IV matrix C_N^{IV} are defined by (4.4) and for values of $N = 2$, 4 and 8, we have the following explicit forms:

$$C_2^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix}, \quad C_4^{\mathrm{IV}} = \frac{1}{\sqrt{2}}\begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix},$$

$$C_8^{\mathrm{IV}} = \frac{1}{2}\begin{pmatrix} \cos\frac{\pi}{32} & \cos\frac{3\pi}{32} & \cos\frac{5\pi}{32} & \cos\frac{7\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{5\pi}{32} & \sin\frac{3\pi}{32} & \sin\frac{\pi}{32} \\ \cos\frac{3\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{\pi}{32} & -\sin\frac{5\pi}{32} & -\cos\frac{5\pi}{32} & -\cos\frac{\pi}{32} & -\cos\frac{7\pi}{32} & -\sin\frac{3\pi}{32} \\ \cos\frac{5\pi}{32} & \sin\frac{\pi}{32} & -\cos\frac{7\pi}{32} & -\cos\frac{3\pi}{32} & -\sin\frac{3\pi}{32} & \sin\frac{7\pi}{32} & \cos\frac{\pi}{32} & \sin\frac{5\pi}{32} \\ \cos\frac{7\pi}{32} & -\sin\frac{5\pi}{32} & -\cos\frac{3\pi}{32} & \sin\frac{\pi}{32} & \cos\frac{\pi}{32} & \sin\frac{3\pi}{32} & -\cos\frac{5\pi}{32} & -\sin\frac{7\pi}{32} \\ \sin\frac{7\pi}{32} & -\cos\frac{5\pi}{32} & -\sin\frac{3\pi}{32} & \cos\frac{\pi}{32} & -\sin\frac{\pi}{32} & -\cos\frac{3\pi}{32} & \sin\frac{5\pi}{32} & \cos\frac{7\pi}{32} \\ \sin\frac{5\pi}{32} & -\cos\frac{\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{3\pi}{32} & -\cos\frac{3\pi}{32} & \cos\frac{7\pi}{32} & \sin\frac{\pi}{32} & -\cos\frac{5\pi}{32} \\ \sin\frac{3\pi}{32} & -\cos\frac{7\pi}{32} & \cos\frac{\pi}{32} & -\cos\frac{5\pi}{32} & \sin\frac{5\pi}{32} & \sin\frac{\pi}{32} & -\sin\frac{7\pi}{32} & \cos\frac{3\pi}{32} \\ \sin\frac{\pi}{32} & -\sin\frac{3\pi}{32} & \sin\frac{5\pi}{32} & -\sin\frac{7\pi}{32} & \cos\frac{7\pi}{32} & -\cos\frac{5\pi}{32} & \cos\frac{3\pi}{32} & -\cos\frac{\pi}{32} \end{pmatrix}.$$

Investigating the basis functions or basis vectors, i.e., rows of the matrices C_{N+1}^{I}, S_{N-1}^{I}, $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$ and C_N^{II}, we observe that they exhibit certain symmetries. The location of symmetry center is determined by the length or order of the row, which is the discrete number of elements. When the order of the matrix is even, the symmetry center is located in midpoint between adjacent center elements of the row vector. Denoting the elements of the k-th row of these matrices by Ψ_{kn} the basis vector is said to be symmetric, if $\Psi_{k,N-1-n} = \Psi_{kn}$ and antisymmetric, if $\Psi_{k,N-1-n} = -\Psi_{kn}$, $n = 0, 1, \ldots, \frac{N}{2} - 1$. Observing $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$ and C_N^{II} matrices we see that the even-indexed rows are symmetric basis vectors, while odd-indexed rows are antisymmetric basis vectors. Indeed, substituting $n = N - 1 - n$ for n into (4.2), (4.9) and (4.10) we note

$$\Psi_{k,N-1-n} = (-1)^k \Psi_{kn}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.15}$$

or

$$\Psi_{2k,N-1-n} = \Psi_{2k,n}, \quad \Psi_{2k+1,N-1-n} = -\Psi_{2k+1,n}, \quad k,n = 0,1,\ldots,\frac{N}{2}-1. \tag{4.16}$$

Matrices with an even order exhibiting such symmetry denoted in general by T_N, may be, after proper bit-reversal of the rows and/or column indices, factorized into sparse matrices as follows [37]:

$$T_N = \begin{pmatrix} E_{\frac{N}{2}} & \bar{E}_{\frac{N}{2}} \\ O_{\frac{N}{2}} & -\bar{O}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} E_{\frac{N}{2}} & 0 \\ 0 & \bar{O}_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.17}$$

or alternatively as [55]

$$T_N = \begin{pmatrix} E_{\frac{N}{2}} & \bar{E}_{\frac{N}{2}} \\ O_{\frac{N}{2}} & -\bar{O}_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} E_{\frac{N}{2}} & 0 \\ 0 & O_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ I_{\frac{N}{2}} & -J_{\frac{N}{2}} \end{pmatrix}, \tag{4.18}$$

where $E_{\frac{N}{2}}$ and $O_{\frac{N}{2}}$ are matrices of order $\frac{N}{2}$ with symmetric and antisymmetric properties as defined in (4.15) and (4.16). $\bar{E}_{\frac{N}{2}}$ and $\bar{O}_{\frac{N}{2}}$ are given by

$$\bar{E}_{\frac{N}{2}} = E_{\frac{N}{2}} J_{\frac{N}{2}}, \quad \bar{O}_{\frac{N}{2}} = O_{\frac{N}{2}} J_{\frac{N}{2}}.$$

When the order of the matrix is odd, the symmetry center is located on the center element of the row vector. Whether a row vector is symmetric or antisymmetric is again in correspondence with the row index, according to

$$\Psi_{k,N-1} = (-1)^k \Psi_{kn}, \quad n = 0,1,\ldots,\frac{N}{2}-1, \tag{4.19}$$

or

$$\Psi_{2k,N-1} = \Psi_{2k,n}, \quad \Psi_{2k+1,N-1} = -\Psi_{2k+1,n}, \quad k,n = 0,1,\ldots,\frac{N}{2}-1. \tag{4.20}$$

Sparse matrix factorization is possible for an odd-order matrix similar to (4.17) and (4.18), except for the "center" elements in the rows and columns.

Transform matrices of such symmetries, such as C^{I}_{N+1}, S^{I}_{N-1}, $\tilde{C}^{\mathrm{I}}_{N}$, $\tilde{S}^{\mathrm{I}}_{N}$ and C^{II}_{N}, are said to form a class of even/odd transforms (EOT). The symmetry and antisymmetry inherent in the basis vectors of the DCT and DST matrices are essential in manipulating and factorizing these transform matrices.

4.4 The fast rotation-based DCT/DST algorithms

In this section, we review the fast and numerically stable DCT/DST algorithms (radix-2 and split-radix) with regular structure and employing real arithmetic only. Almost all are direct

algorithms based on real factorization of the corresponding DCT/DST matrix into products of sparse (orthogonal) matrices of simple structure. Most of them are completely recursive and use only permutations, butterfly operations and rotations. Compared to direct matrix–vector multiplication for a given N-length input data vector, the fast DCT/DST algorithms of radix-2 reduce the computational complexity approximately from $2N^2$ arithmetic operations (N^2 multiplications and $N(N-1)$ additions) to $2N\log_2 N$ arithmetic operations ($\frac{N}{2}\log_2 N$ multiplications and $N\log_2 N$ additions).

The fast algorithms for a specific DCT/DST are presented in the order as were reported in the literature. The description of each fast algorithm will include:

- General comments with reference to the bibliography,
- Complete formulae and/or
- Recursive (if there exist) sparse matrix factorizations to generate higher-order transform matrices from lower-order ones,
- The corresponding regular generalized signal flow graphs involving transform block sizes $N = 2$, 4 and 8.

We note that symmetric transform matrices C_{N+1}^{I}, S_{N-1}^{I}, $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$, C_N^{IV} and S_N^{IV} imply that the forward and inverse computational algorithms are identical, and the sparse matrix factorization will have the alternative transposed versions.

4.4.1 The fast DCT-I and SCT algorithms

4.4.1.1 DCT-I computation based on the SCT algorithm for $N = 2^m + 1$

For the efficient SCT (see (4.9)) computation, a fast recursive algorithm with regular structure has been proposed [10]. Assuming $N = 2^m + 1$, the SCT algorithm can be adapted for a new fast recursive computation of the DCT-I as follows.

Using $\tilde{c}_k^{\mathrm{I}}$ and x_n to represent the k-th and n-th elements of the transformed vector and input vector, respectively, (4.9) for $M = N - 1$ can be expressed as

$$\tilde{c}_k^{\mathrm{I}} = \sqrt{\frac{2}{M}}\epsilon_k \sum_{n=0}^{M} \epsilon_n x_n \cos\frac{\pi nk}{M}, \quad k = 0, 1, \ldots, M.$$

Ignoring the scaling factors, the essential part of the above sum can be expressed as

$$\tilde{c}_k^{\mathrm{I}} = \sum_{n=0}^{M-1} x_n \cos\frac{\pi nk}{M} + (-1)^k x_M = a_M(k) + (-1)^k x_M, \quad k = 0, 1, \ldots, M.$$

Splitting the sum $a_M(k)$ into even-indexed and odd-indexed points (i.e., respectively grouping $2n$ and $2n+1$ terms) and applying the symmetries of transform kernels the complete

formulae are given by

$$a_M(k) = a_{\frac{M}{2}}(k) + b_{\frac{M}{2}}(k), \quad k = 0, 1, \ldots, \frac{M}{2}, \quad b_{\frac{M}{2}}\left(\frac{M}{2}\right) = 0,$$
$$a_M(M-k) = a_{\frac{M}{2}}(k) - b_{\frac{M}{2}}(k), \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \tag{4.21}$$

where

$$a_{\frac{M}{2}}(k) = \sum_{n=0}^{\frac{M}{2}-1} x_{2n} \cos \frac{\pi nk}{M/2},$$
$$b_{\frac{M}{2}}(k) = \sum_{n=0}^{\frac{M}{2}-1} x_{2n+1} \cos \frac{\pi(2n+1)k}{2(M/2)}. \tag{4.22}$$

The $(M+1)$-point DCT-I is recursively decomposed into $(\frac{M}{2}+1)$-point DCT-I and $\frac{M}{2}$-point DCT-II. The corresponding generalized signal flow graph for the forward and inverse DCT-I computation for $N = 3$, 5 and 9 is shown in Fig. 4.1. The input data sequence $\{x_n\}$ is in bit-reversed order. Full lines in Fig. 4.1 represent unity transfer factors while broken lines represent transfer factors -1. $\bigcirc$ represents addition and $\downarrow$ represents multiplication after addition. These notations hold through the chapter for all signal flow graphs.

On the other hand, the N-point SCT defined by (4.9) for $N = 2^n$, without the scaling factors can be expressed as

$$\tilde{c}_k^{\mathrm{I}} = \sum_{n=0}^{N-1} x_n \cos \frac{\pi nk}{N-1} = a_N(k), \quad k = 0, 1, \ldots, N-1,$$

and decomposed by similar method used in (4.21) and (4.22) into $\frac{N}{2}$-point SCT and $\frac{N}{2}$-point DCT-II as follows

$$a_N(k) = a_{\frac{N}{2}}(k) + b_{\frac{N}{2}}(k), \quad k = 0, 1, \ldots, \frac{N}{2}-1,$$
$$a_N(N-1-k) = a_{\frac{N}{2}}(k) - b_{\frac{N}{2}}(k), \quad k = 0, 1, \ldots, \frac{N}{2}-1, \tag{4.23}$$

where

$$a_{\frac{N}{2}}(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \cos \frac{\pi nk}{N-1},$$
$$b_{\frac{N}{2}}(k) = \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \cos \frac{\pi(2n+1)k}{N-1}. \tag{4.24}$$

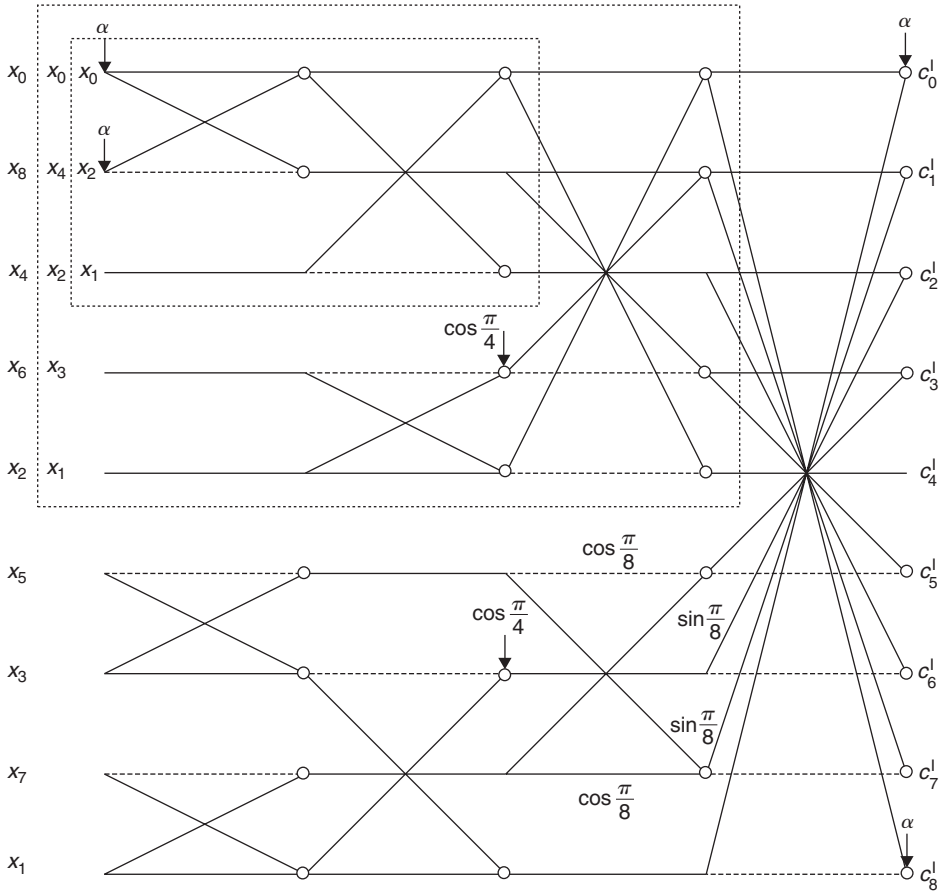


Fig. 4.1. The generalized signal flow graph for the forward and inverse DCT-I computation for $N = 3$, 5 and 9 based on the SCT algorithm; $\alpha = \frac{\sqrt{2}}{2}$.

Unfortunately, the decomposition of the SCT given by (4.23) and (4.24) is not recursive implying that the SCT matrix $\tilde{C}_N^{\mathrm{I}}$ does not have a recursive structure. Alternatively, since SCT is an EOT the $\tilde{C}_N^{\mathrm{I}}$ matrix can be non-recursively factorized according to (4.17).

4.4.1.2 DCT-I recursive sparse matrix factorizations

The DCT-I matrix C_{N+1}^{I} for $N = 2^m$ can be factorized into the following recursive sparse matrix form [7, 32, 40]:

$$C_{N+1}^{\mathrm{I}} = P_{N+1} \begin{pmatrix} C_{\frac{N}{2}+1}^{\mathrm{I}} & 0 \\ 0 & J_{\frac{N}{2}} C_{\frac{N}{2}}^{\mathrm{III}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ J_{\frac{N}{2}} & & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.25}$$

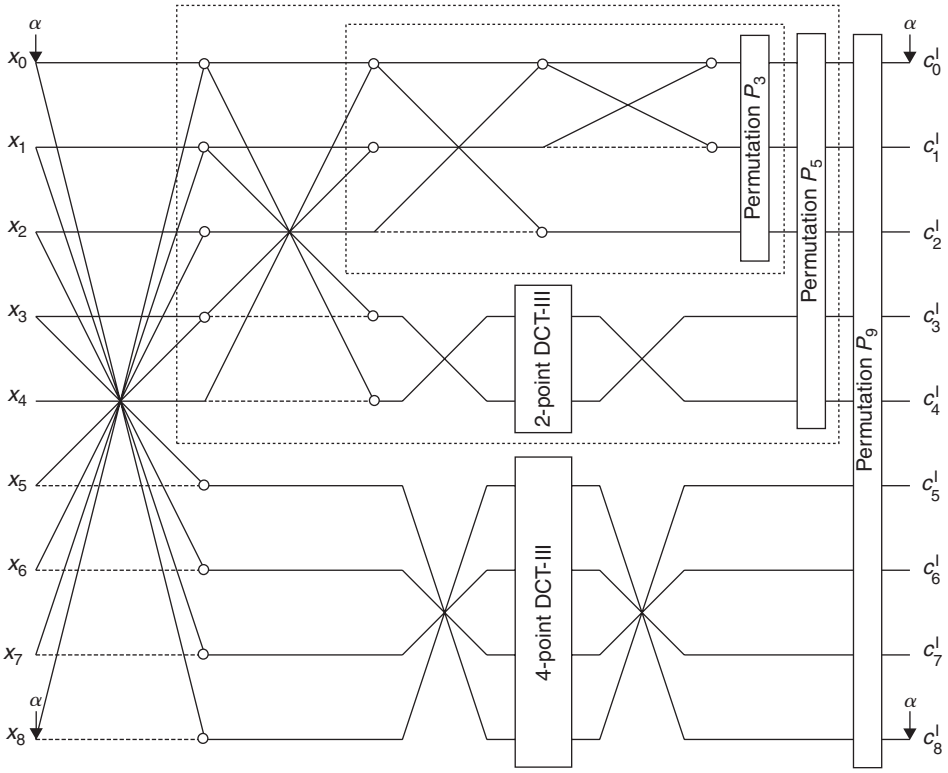


Fig. 4.2. The generalized signal flow graph for the forward and inverse DCT-I computation for $N = 2$, 4 and 8 based on recursive sparse matrix factorization (4.25); $\alpha = \frac{\sqrt{2}}{2}$.

where P_{N+1} is a permutation matrix, when it is applied to a data vector it corresponds to the reordering

$$\tilde{x}_0 = x_0, \;\; \tilde{x}_{n+1} = x_{2n+2}, \;\; \tilde{x}_{N-n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.26}$$

The $(N+1)$-point DCT-I is decomposed recursively into $(\frac{N}{2}+1)$-point DCT-I and $\frac{N}{2}$-point DCT-III. The generalized signal flow graph for the forward and inverse DCT-I computation for $N = 2$, 4 and 8 is shown in Fig. 4.2.

Similarly, an orthogonal recursive sparse matrix factorization of the DCT-I matrix C^{I}_{N+1} with scaling $\sqrt{2}$ has been introduced in Ref. [55]. For $N = 2^m$, $m > 1$, the matrix C^{I}_{N+1} can be factorized in the form:

$$C^{\mathrm{I}}_{N+1} = P^{\mathrm{T}}_{N+1} \begin{pmatrix} C^{\mathrm{I}}_{\frac{N}{2}+1} & 0 \\ 0 & C^{\mathrm{III}}_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ I_{\frac{N}{2}} & & -J_{\frac{N}{2}} \end{pmatrix}$$

$$= P_{N+1}^{\mathrm{T}} \begin{pmatrix} C_{\frac{N}{2}+1}^{\mathrm{I}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{III}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & & 0 \\ & 1 & \\ 0 & & J_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ J_{\frac{N}{2}} & & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.27}$$

where P_{N+1} is a permutation matrix effecting the reordering

$$\tilde{x}_0 = x_0, \;\; \tilde{x}_{n+1} = x_{2n+2}, \;\; \tilde{x}_{\frac{N}{2}+1+n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.28}$$

Note that $P_{N+1}^{\mathrm{T}} = P_{N+1}^{-1}$, here.

4.4.1.3 The split-radix DCT-I algorithm

The idea of 2^m split-radix fast Fourier transform (FFT) algorithm [24–26] was extended to the DCT-I [27]. The complete formulae of split-radix fast DCT-I algorithm (normalization factors are omitted) are given by

$$\begin{aligned} c_{2k}^{\mathrm{I}} &= \sum_{n=0}^{\frac{N}{2}} (x_n + x_{N-n}) \cos \frac{\pi nk}{N/2} - x_{\frac{N}{2}} \cos \pi k, \quad k = 0, 1, \ldots, \frac{N}{2}, \\ c_{4k-1}^{\mathrm{I}} &= a_k + b_k, \quad k = 1, 2, \ldots, \frac{N}{4}, \\ c_{4k+1}^{\mathrm{I}} &= a_k - b_k, \quad k = 0, 1, \ldots, \frac{N}{4} - 1, \;\; b_0 = 0, \end{aligned} \tag{4.29}$$

where

$$\begin{aligned} a_k &= \sum_{n=0}^{\frac{N}{4}} \left[(x_n - x_{N-n}) \cos \frac{\pi n}{N} + (x_{\frac{N}{2}-n} - x_{\frac{N}{2}+n}) \sin \frac{\pi n}{N} \right] \cos \frac{\pi nk}{N/4} \\ &\quad - (x_{\frac{N}{4}} - x_{\frac{3N}{4}}) \cos \frac{\pi}{4}(4k+1), \\ b_k &= \sum_{n=1}^{\frac{N}{4}-1} \left[(x_n - x_{N-n}) \sin \frac{\pi n}{N} - (x_{\frac{N}{2}-n} - x_{\frac{N}{2}+n}) \cos \frac{\pi n}{N} \right] \sin \frac{\pi nk}{N/4}. \end{aligned} \tag{4.30}$$

Thus, the first stage of split-radix decomposition replaces $(N+1)$-point DCT-I by one $(\frac{N}{2}+1)$-point DCT-I, one DCT-I of length $(\frac{N}{4}+1)$ and one DST-I of length $(\frac{N}{4}-1)$. The decomposition is used recursively. It results in the generalized signal flow graph with regular structure which is shown for $N = 2$, 4 and 8 in Fig. 4.3. The output data sequence $\{c_k^{\mathrm{I}}\}$ is in bit-reversed order. It is interesting to compare the generalized signal flow graphs

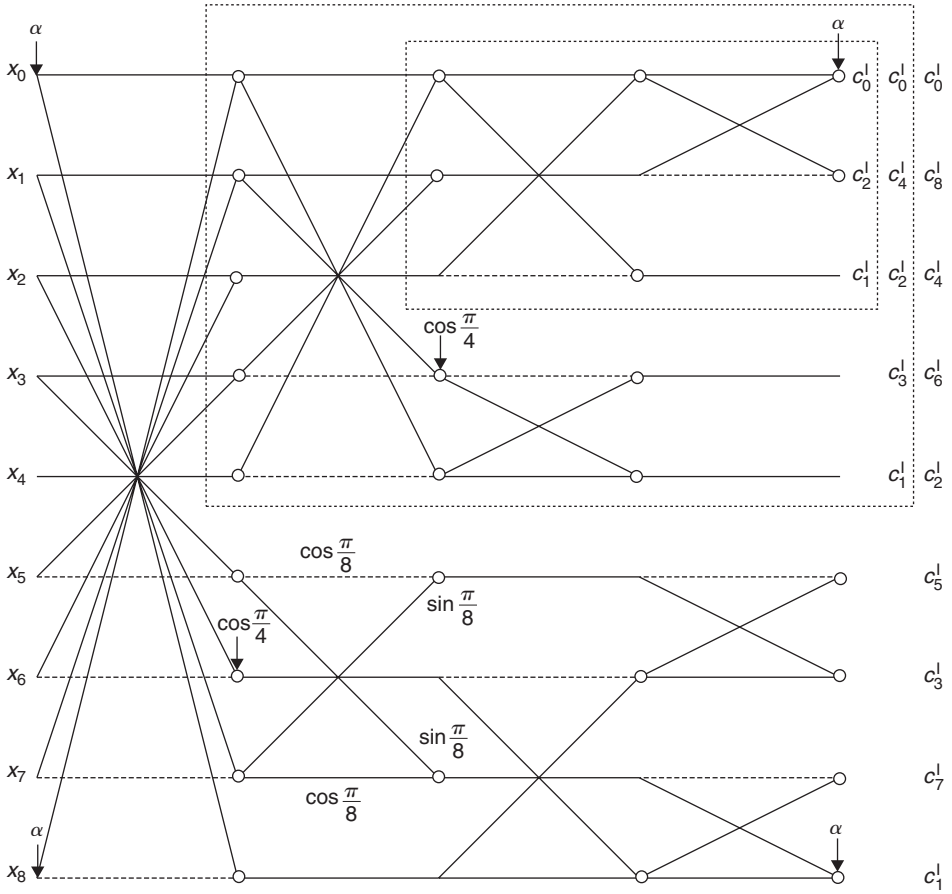


Fig. 4.3. The generalized signal flow graph for the forward and inverse DCT-I computation for $N = 2$, 4 and 8 based on the split-radix algorithm; $\alpha = \frac{\sqrt{2}}{2}$.

shown in Figs. 4.1 and 4.3. Their lower parts imply that there exists a factorization of $\frac{N}{2}$-point DCT-III (see the factorizations given by (4.25) and (4.27)) consisting of a rotation matrix, $(\frac{N}{4}+1)$-point DCT-I and $(\frac{N}{4}-1)$-point DST-I whose outputs are combined in the final stage. Such a factorization of the C_N^{III} matrix actually exists (see (4.56) in Section 4.4.3.1). Moreover, these generalized signal flow graphs are mirror images of each other although they are derived from quite differently formulated algorithms.

According to split-radix DCT-I algorithm, the matrix C_{N+1}^{I} can be recursively factorized as follows

$$C_{N+1}^{\mathrm{I}} = P_{N+1} \begin{pmatrix} C_{\frac{N}{2}+1}^{\mathrm{I}} & 0 \\ 0 & K_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ J_{\frac{N}{2}} & & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.31}$$

where

$$K_{\frac{N}{2}} = \begin{pmatrix} -I_{\frac{N}{4}-1} & J_{\frac{N}{4}-1} & 0 \\ J_{\frac{N}{4}-1} & I_{\frac{N}{4}-1} & 0 \\ 0 & 0 & I_2 \end{pmatrix} \begin{pmatrix} \bar{S}^{\mathrm{I}}_{\frac{N}{4}-1} & 0 \\ 0 & \bar{C}^{\mathrm{I}}_{\frac{N}{4}+1} \end{pmatrix} R_{\frac{N}{2}}, \tag{4.32}$$

and

$$K_1 = \frac{1}{\sqrt{2}}, \quad K_2 = \begin{pmatrix} -\cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \end{pmatrix}, \quad C^{\mathrm{I}}_2 = \frac{1}{2}\begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}. \tag{4.33}$$

P_{N+1} is a permutation matrix for reordering from bit-reversal to natural order. $\bar{S}^{\mathrm{I}}_{\frac{N}{4}-1}$ and $\bar{C}^{\mathrm{I}}_{\frac{N}{4}+1}$ are unnormalized DST-I and DCT-I matrices respectively given by

$$\bar{S}^{\mathrm{I}}_{\frac{N}{4}-1} = J_{\frac{N}{4}-1}(B_{\frac{N}{4}-1}S^{\mathrm{I}}_{\frac{N}{4}-1})J_{\frac{N}{4}-1},$$
$$\bar{C}^{\mathrm{I}}_{\frac{N}{4}+1} = J_{\frac{N}{4}+1}(B_{\frac{N}{4}+1}C^{\mathrm{I}}_{\frac{N}{4}+1})J_{\frac{N}{4}+1},$$

where $B_{\frac{N}{4}-1}$ and $B_{\frac{N}{4}+1}$ are bit-reversal permutation matrices. $R_{\frac{N}{2}}$ is a rotation matrix given by

$$R_{\frac{N}{2}} = \begin{pmatrix} -\cos\frac{\pi}{N} & & & & 0 & & & & \sin\frac{\pi}{N} & 0 \\ & -\cos\frac{2\pi}{N} & & & & & & \sin\frac{2\pi}{N} & & \\ & & \ddots & & & & \iddots & & & \\ & & & -\cos\frac{(\frac{N}{4}-1)\pi}{N} & & \sin\frac{(\frac{N}{4}-1)\pi}{N} & & & & \\ 0 & & & & \cos\frac{\pi}{4} & & & & & 0 \\ & & & \sin\frac{(\frac{N}{4}-1)\pi}{N} & & \cos\frac{(\frac{N}{4}-1)\pi}{N} & & & & \\ & & \iddots & & & & \ddots & & & \\ & \sin\frac{2\pi}{N} & & & & & & \cos\frac{2\pi}{N} & & \\ \sin\frac{\pi}{N} & & & & & & & & \cos\frac{\pi}{N} & 0 \\ 0 & & & & 0 & & & & 0 & \frac{1}{\sqrt{2}} \end{pmatrix}.$$

We note that the recursive sparse matrix factorization of DCT-I matrix given by (4.31)–(4.33) is valid for $N = 2$, 4 and 8.

On the other hand, the matrix C^{I}_{N+1} can be recursively factorized into $C^{\mathrm{I}}_{\frac{N}{2}+1}$ and $C^{\mathrm{III}}_{\frac{N}{2}}$, and by orthogonal recursive matrix factorizations given by (4.27) and (4.56), we can obtain the orthogonal factorization of C^{I}_{N+1} as [55]

$$C^{\mathrm{I}}_{N+1} = P^{\mathrm{T}}_{N+1}\begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & P^{\mathrm{T}}_{\frac{N}{2}}A_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} C^{\mathrm{I}}_{\frac{N}{2}+1} & & \\ & C^{\mathrm{I}}{}_{\frac{N}{4}+1} & \\ & & S^{\mathrm{I}}_{\frac{N}{4}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & T^{(0)}_{\frac{N}{2}} \end{pmatrix} T^{(1)}_{N+1}, \tag{4.34}$$

with orthogonal matrices

$$A_{\frac{N}{2}} = \begin{pmatrix} I_{\frac{N}{4}} & 0 \\ 0 & J_{\frac{N}{4}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & \\ & I_{\frac{N}{4}-1} & & J_{\frac{N}{4}-1} \\ & & \sqrt{2} & \\ & J_{\frac{N}{4}-1} & & -I_{\frac{N}{4}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{4}+1} & 0 \\ 0 & (-1)^{\frac{N}{4}} D_{\frac{N}{4}-1} \end{pmatrix},$$

$$T^{(1)}_{N+1} = \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ I_{\frac{N}{2}} & & -J_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} I_{\frac{N}{2}} & & \\ & 1 & \\ & & J_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & \sqrt{2} & \\ I_{\frac{N}{2}} & & -J_{\frac{N}{2}} \end{pmatrix},$$

and

$$T^{(0)}_{\frac{N}{2}} = \begin{pmatrix} I_{\frac{N}{4}+1} & 0 \\ 0 & D_{\frac{N}{4}-1} \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & & & & 0 & & & 0 \\ 0 & \cos\frac{\pi}{2N} & & & & & & & \sin\frac{\pi}{2N} \\ & & \cos\frac{2\pi}{2N} & & & & & \sin\frac{2\pi}{2N} & \\ & & & \ddots & & & \iddots & & \\ & & & & \cos\frac{(\frac{N}{4}-1)\pi}{2N} & & \sin\frac{(\frac{N}{4}-1)\pi}{2N} & & \\ 0 & & & & & 1 & & & 0 \\ & & & & -\sin\frac{(\frac{N}{4}-1)\pi}{2N} & & \cos\frac{(\frac{N}{4}-1)\pi}{2N} & & \\ & & & \iddots & & & \ddots & & \\ & & -\sin\frac{2\pi}{2N} & & & & & \cos\frac{2\pi}{2N} & \\ 0 & -\sin\frac{\pi}{2N} & & & & 0 & & & \cos\frac{\pi}{2N} \end{pmatrix},$$

where P_{N+1} is a permutation matrix defined by (4.26), and permutation matrix $P_{\frac{N}{2}}$ is defined by (4.54). $D_{\frac{N}{4}-1} = \mathrm{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{4} - 2$ is the diagonal odd-sign changing matrix.

4.4.2 The fast DST-I and SST algorithms

4.4.2.1 DST-I computation based on the SST algorithm for $N = 2^m - 1$

Following the derivation of fast SCT algorithm [10] we can derive by similar methods a new fast algorithm for SST (see (4.10)) computation with regular structure. Assuming $N = 2^m - 1$, it can be adapted for the new fast recursive computation of the DST-I as follows.

Rewriting (4.10) for $M = N + 1$ we have

$$\tilde{s}^{\mathrm{I}}_k = \sqrt{\frac{2}{M}} \sum_{n=0}^{M} x_n \sin\frac{\pi(n+1)(k+1)}{M} = a_M(k), \quad k = 0, 1, \ldots, M-2.$$

Ignoring the scaling factor we can split the sum $a_M(k)$ into even-indexed and odd-indexed points (i.e., respectively grouping $2n$ and $2n+1$ terms) and using the symmetries of transform kernels, the complete formulae are given by

$$\begin{aligned} a_M(k) &= b_{\frac{M}{2}}(k) + a_{\frac{M}{2}}(k), \quad k = 0, 1, \ldots, \frac{M}{2} - 1, \;\; a_{\frac{M}{2}}\left(\frac{M}{2} - 1\right) = 0, \\ a_M(M-k-2) &= b_{\frac{M}{2}}(k) - a_{\frac{M}{2}}(k), \quad k = 0, 1, \ldots, \frac{M}{2} - 2, \end{aligned} \tag{4.35}$$

where

$$\begin{aligned} a_{\frac{M}{2}}(k) &= \sum_{n=0}^{\frac{M}{2}-2} x_{2n+1} \sin \frac{\pi(n+1)(k+1)}{M/2}, \\ b_{\frac{M}{2}}(k) &= \sum_{n=0}^{\frac{M}{2}-1} x_{2n} \sin \frac{\pi(2n+1)(k+1)}{2(M/2)}. \end{aligned} \tag{4.36}$$

Hence, the $(M-1)$-point DST-I is recursively decomposed into an $\frac{M}{2}$-point DST-II and an $(\frac{M}{2}-1)$-point DST-I. The corresponding generalized signal flow graph for the forward and inverse DST-I computation for $N = 3$ and 7 is shown in Fig. 4.4. The input data sequence $\{x_n\}$ is in bit-reversed order. The output data sequence $\{s_k^{\mathrm{I}}\}$ is in reverse order.

On the other hand, the N-point SST defined by (4.10) for $N = 2^n$, without the scaling factors can be expressed as

$$\tilde{s}_k^{\mathrm{I}} = \sum_{n=0}^{N-1} x_n \sin \frac{\pi(n+1)(k+1)}{N+1} = a_N(k), \quad k = 0, 1, \ldots, N-1,$$

and decomposed by similar methods used in (4.35) and (4.36) into $\frac{N}{2}$-point SST and $\frac{N}{2}$-point DST-II as follows:

$$\begin{aligned} a_N(k) &= b_{\frac{N}{2}}(k) + a_{\frac{N}{2}}(k), \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \\ a_N(N-k-1) &= b_{\frac{N}{2}}(k) - a_{\frac{N}{2}}(k), \quad k = 0, 1, \ldots, \frac{N}{2} - 1, \end{aligned} \tag{4.37}$$

where

$$\begin{aligned} a_{\frac{N}{2}}(k) &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n+1} \sin \frac{\pi(n+1)(k+1)}{N+1}, \\ b_{\frac{N}{2}}(k) &= \sum_{n=0}^{\frac{N}{2}-1} x_{2n} \sin \frac{2\pi(2n+1)(k+1)}{N+1}. \end{aligned} \tag{4.38}$$

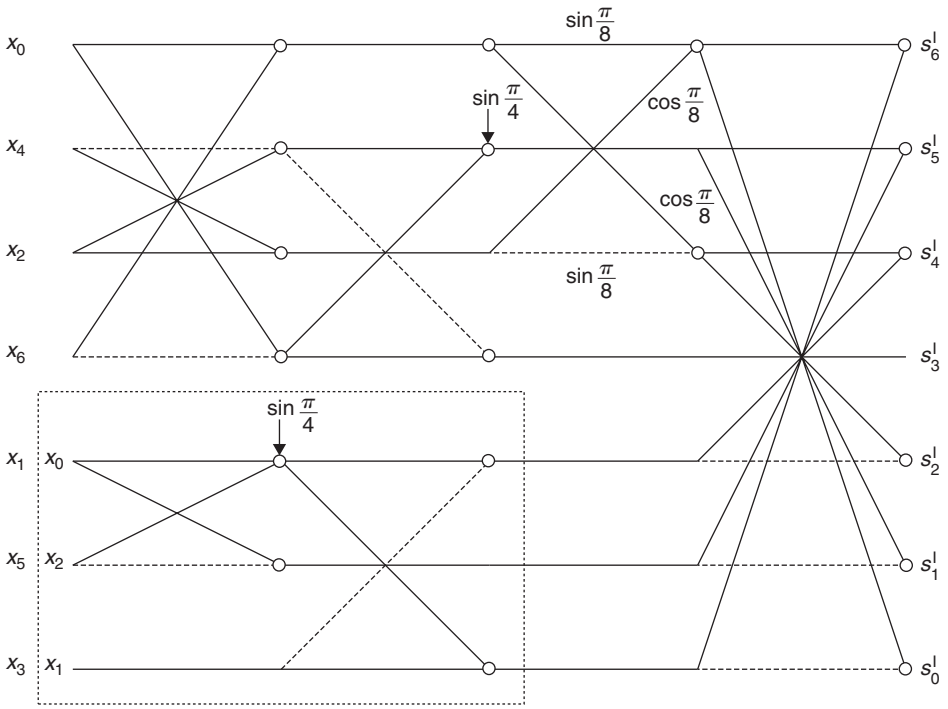


Fig. 4.4. The generalized signal flow graph for the forward and inverse DST-I computation for $N = 3$ and 7 based on the SST algorithm.

Unfortunately, the decomposition of the SST given by (4.37) and (4.38) is not recursive implying that the SST matrix $\tilde{S}_N^{\mathrm{I}}$ does not have a recursive structure. Alternatively, since SST is an EOT the $\tilde{S}_N^{\mathrm{I}}$ matrix can be non-recursively factorized according to (4.17).

4.4.2.2 DST-I recursive sparse matrix factorizations

One of the first recursive sparse matrix factorizations of the DST-I matrix was reported in Ref. [29] and its corrected forms for $N = 8$ in Refs. [2, 30, 31]. The modified recursive sparse matrix factorization of the DST-I matrix for $N = 2^m$ is defined as

$$S_{N-1}^{\mathrm{I}} = B_{N-1} \begin{pmatrix} B_{\frac{N}{2}} S_{\frac{N}{2}}^{\mathrm{III}} & 0 \\ 0 & S_{\frac{N}{2}-1}^{\mathrm{I}} J_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix}, \tag{4.39}$$

where B_{N-1} is a permutation matrix permuting the transformed data sequence from the bit-reversal order to natural order. The corresponding generalized signal flow graph for $N = 4$ and 8 is shown in Fig. 4.5.

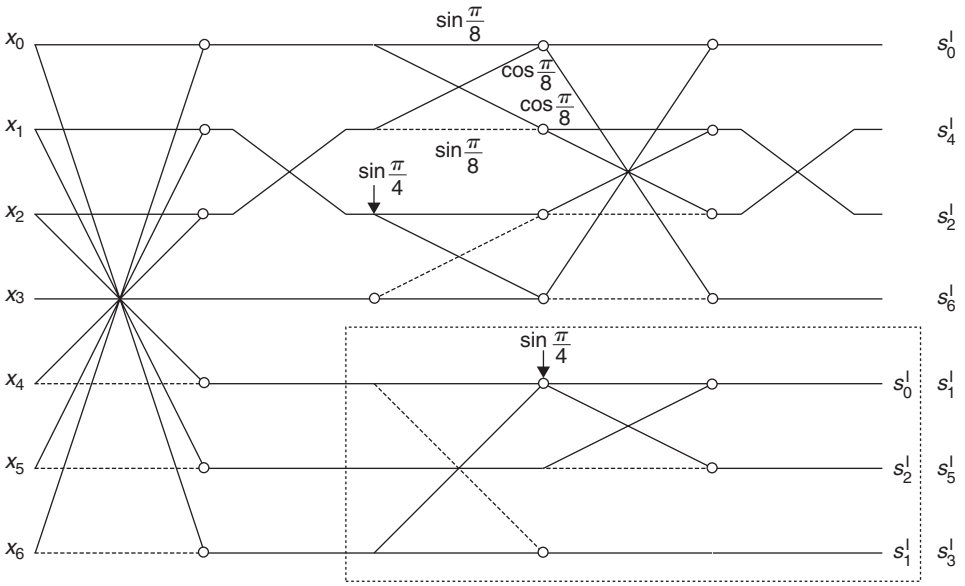


Fig. 4.5. The generalized signal flow graph for the forward and inverse DST-I computation for $N = 4$ and 8 based on (4.39).

A slightly different alternative recursive sparse matrix factorization of the DST-I matrix for $N = 2^m$ is defined as [7, 32, 40]

$$S^{\mathrm{I}}_{N-1} = P_{N-1} \begin{pmatrix} S^{\mathrm{III}}_{\frac{N}{2}} & 0 \\ 0 & J_{\frac{N}{2}-1} S^{\mathrm{I}}_{\frac{N}{2}-1} J_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix}, \tag{4.40}$$

where P_{N-1} is a permutation matrix which when applied to a data vector corresponds to the reordering

$$\tilde{x}_0 = x_0, \;\; \tilde{x}_{n+1} = x_{2n+2}, \;\; \tilde{x}_{N-2-n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 2. \tag{4.41}$$

The generalized signal flow graph for the forward and inverse DST-I computation for $N = 4$ and 8 is shown in Fig. 4.6. In both factorizations (4.39) and (4.40), the $(N-1)$-point DST-I is recursively decomposed into $\frac{N}{2}$-point DST-III and $(\frac{N}{2}-1)$-point DST-I.

Similarly, an orthogonal recursive sparse matrix factorization of the DST-I matrix S^{I}_{N-1} with scaling $\sqrt{2}$ has been introduced in Ref. [55]. For $N = 2^m, m > 1$, the matrix S^{I}_{N-1} can

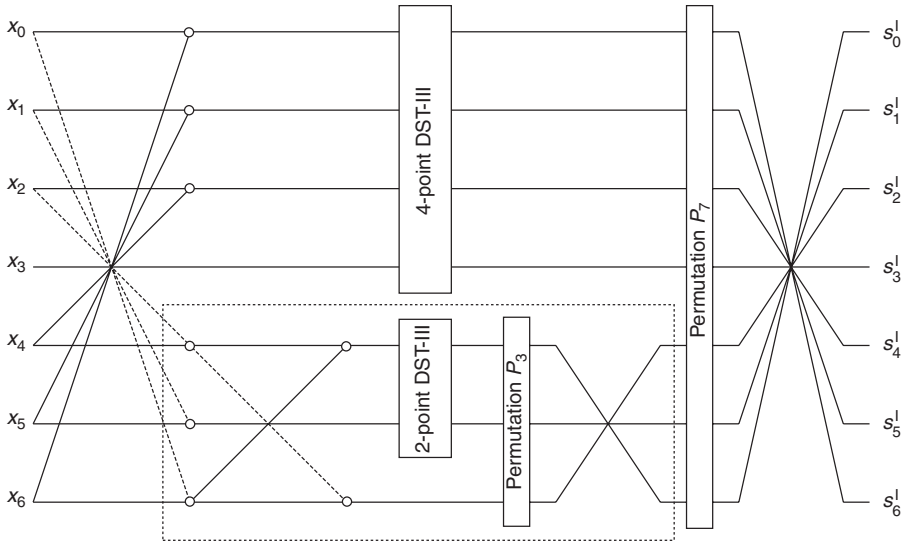


Fig. 4.6. The generalized signal flow graph for the forward and inverse DST-I computation for $N = 4$ and 8 based on (4.40).

be factorized in the form:

$$S^{\mathrm{I}}_{N-1} = P^{\mathrm{T}}_{N-1} \begin{pmatrix} S^{\mathrm{III}}_{\frac{N}{2}} & 0 \\ 0 & S^{\mathrm{I}}_{\frac{N}{2}-1} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ I_{\frac{N}{2}-1} & & -J_{\frac{N}{2}-1} \end{pmatrix}$$

$$= P^{\mathrm{T}}_{N-1} \begin{pmatrix} S^{\mathrm{III}}_{\frac{N}{2}} & 0 \\ 0 & S^{\mathrm{I}}{}_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}-1} & & \\ & 1 & \\ & & J_{\frac{N}{2}-1} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix}, \tag{4.42}$$

or using the relation (4.11) in the form:

$$S^{\mathrm{I}}_{N-1}$$

$$= P^{\mathrm{T}}_{N-1} \begin{pmatrix} D_{\frac{N}{2}} & 0 \\ 0 & I_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} C^{\mathrm{III}}_{\frac{N}{2}} & 0 \\ 0 & S^{\mathrm{I}}_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} J_{\frac{N}{2}-1} & & \\ & 1 & \\ & & I_{\frac{N}{2}-1} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ I_{\frac{N}{2}-1} & & -J_{\frac{N}{2}-1} \end{pmatrix}$$

$$= P^{\mathrm{T}}_{N-1} \begin{pmatrix} D_{\frac{N}{2}} & 0 \\ 0 & I_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} C^{\mathrm{III}}_{\frac{N}{2}} & 0 \\ 0 & S^{\mathrm{I}}{}_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} J_{\frac{N}{2}-1} & & \\ & 1 & \\ & & J_{\frac{N}{2}-1} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & \sqrt{2} & \\ J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix}, \tag{4.43}$$

where $D_{\frac{N}{2}} = \text{diag}\,\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$ is the diagonal odd-sign changing matrix and P_{N-1} is a permutation matrix which when applied to a data vector corresponds to the reordering

$$\tilde{x}_0 = x_0, \;\; \tilde{x}_{n+1} = x_{2n+2}, \;\; \tilde{x}_{\frac{N}{2}+n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 2. \tag{4.44}$$

Note that $P_{N-1}^{\mathrm{T}} = P_{N-1}^{-1}$, here.

4.4.2.3 The split-radix DST-I algorithm

The idea of 2^m split-radix FFT algorithm [24–26] was also extended to the DST-I [27]. The complete formulae of split-radix fast DST-I algorithm (normalization factors are omitted) are given by

$$\begin{aligned} s_{2k}^{\mathrm{I}} &= \sum_{n=1}^{\frac{N}{2}-1} (x_n - x_{N-n}) \sin \frac{\pi n k}{N/2}, \quad k = 1, \ldots, \frac{N}{2} - 1, \\ s_{4k-1}^{\mathrm{I}} &= a_k - b_k, \quad k = 1, 2, \ldots, \frac{N}{4}, \\ s_{4k+1}^{\mathrm{I}} &= a_k + b_k, \quad k = 0, 1, \ldots, \frac{N}{4} - 1, \;\; a_0 = 0, \end{aligned} \tag{4.45}$$

where

$$\begin{aligned} a_k &= \sum_{n=1}^{\frac{N}{4}-1} \left[(x_n + x_{N-n}) \cos \frac{\pi n}{N} - (x_{\frac{N}{2}-n} + x_{\frac{N}{2}+n}) \sin \frac{\pi n}{N} \right] \sin \frac{\pi n k}{N/4}, \\ b_k &= \sum_{n=0}^{\frac{N}{4}} \left[(x_n + x_{N-n}) \sin \frac{\pi n}{N} + (x_{\frac{N}{2}-n} + x_{\frac{N}{2}+n}) \cos \frac{\pi n}{N} \right] \cos \frac{\pi n k}{N/4} \\ &\quad - x_{\frac{N}{2}} - (x_{\frac{N}{4}} + x_{\frac{3N}{4}}) \cos \frac{\pi}{4}(4k + 1), \end{aligned} \tag{4.46}$$

Hence, the first stage of split-radix decomposition replaces $(N-1)$-point DST-I by one $(\frac{N}{2} - 1)$-point DST-I, one DST-I of length $(\frac{N}{4} - 1)$ and one DCT-I of length $(\frac{N}{4} + 1)$. The decomposition is used recursively. It results in the generalized signal flow graph with regular structure which is shown for $N = 4$ and 8 in Fig. 4.7. The output data sequence $\{s_k^{\mathrm{I}}\}$ is in the bit-reverse order.

Consider the generalized signal flow graph shown in Fig. 4.7. Its upper part implies that there exists a factorization of N-point DST-III (see the factorization given by (4.40)) consisting of a rotation matrix, an $(\frac{N}{2} - 1)$-point DST-I and an $(\frac{N}{2} + 1)$-point DCT-I, whose outputs are combined in the final stage. Specifically, according to split-radix DST-I

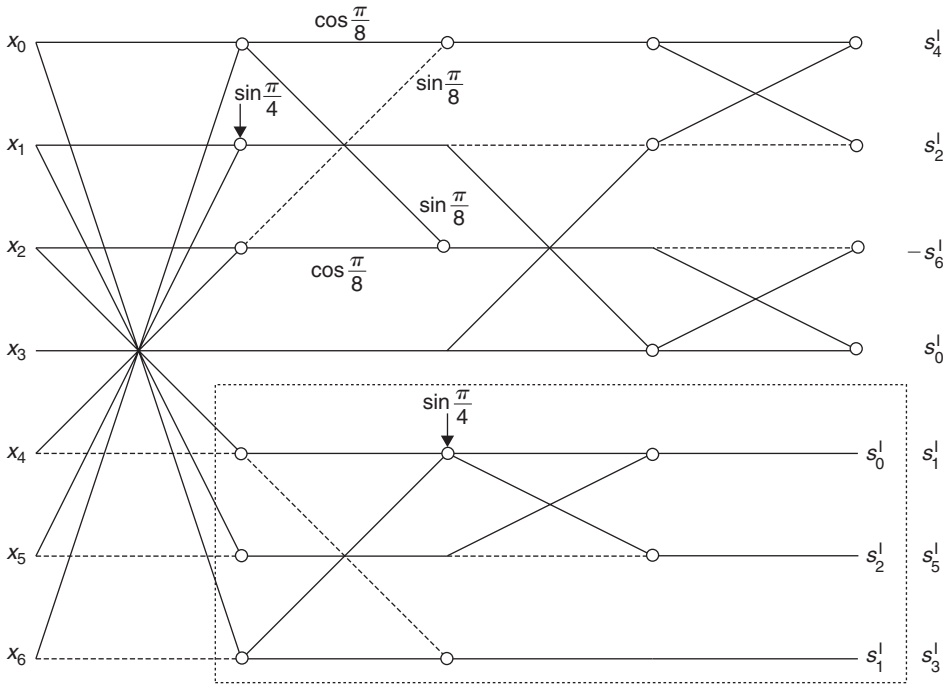


Fig. 4.7. The generalized signal flow graph for the forward and inverse DST-I computation for $N = 4$ and 8 based on split-radix algorithm.

algorithm the matrix S^{I}_{N-1} can be recursively factorized as follows:

$$S^{\mathrm{I}}_{N-1} = P_{N-1} \begin{pmatrix} K_{\frac{N}{2}} & 0 \\ 0 & S^{\mathrm{I}}{}_{\frac{N}{2}-1} J_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & & J_{\frac{N}{2}} \\ & 1 & \\ J_{\frac{N}{2}} & & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.47}$$

where

$$K_{\frac{N}{2}} = Q_{\frac{N}{2}} \begin{pmatrix} I_{\frac{N}{4}-1} & J_{\frac{N}{4}-1} & 0 & 0 \\ J_{\frac{N}{4}-1} & -I_{\frac{N}{4}-1} & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \bar{S}^{\mathrm{I}}_{\frac{N}{4}-1} & 0 \\ 0 & \bar{C}^{\mathrm{I}}_{\frac{N}{4}+1} \end{pmatrix} R_{\frac{N}{2}}, \tag{4.48}$$

and

$$K_2 = \begin{pmatrix} \sin\frac{\pi}{4} & -1 \\ \sin\frac{\pi}{4} & 1 \end{pmatrix} = S^{\mathrm{III}}_2, \quad Q_4 = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}. \tag{4.49}$$

P_{N-1} is a permutation matrix for reordering from bit-reversal to natural order. $\bar{S}^{\mathrm{I}}_{\frac{N}{4}-1}$ and $\bar{C}^{\mathrm{I}}_{\frac{N}{4}+1}$ are unnormalized DST-I and DCT-I matrices respectively given by

$$\bar{S}^{\mathrm{I}}_{\frac{N}{4}-1} = J_{\frac{N}{4}-1}(B_{\frac{N}{4}-1}S^{\mathrm{I}}_{\frac{N}{4}-1})J_{\frac{N}{4}-1},$$

$$\bar{C}^{\mathrm{I}}_{\frac{N}{4}+1} = J_{\frac{N}{4}+1}(B_{\frac{N}{4}+1}C^{\mathrm{I}}_{\frac{N}{4}+1})J_{\frac{N}{4}+1},$$

where $B_{\frac{N}{4}-1}$ and $B_{\frac{N}{4}+1}$ are bit-reversal permutation matrices. $R_{\frac{N}{2}}$ is a rotation matrix given by

$$R_{\frac{N}{2}} = \begin{pmatrix}
\cos\frac{\pi}{N} & & & & 0 & & & & -\sin\frac{\pi}{N} & 0 \\
& \cos\frac{2\pi}{N} & & & & & & -\sin\frac{2\pi}{N} & & \\
& & \ddots & & & & \iddots & & & \\
& & & \cos\frac{(\frac{N}{4}-1)\pi}{N} & & -\sin\frac{(\frac{N}{4}-1)\pi}{N} & & & & \\
0 & & & & \sin\frac{\pi}{4} & & & & & 0 \\
& & & \sin\frac{(\frac{N}{4}-1)\pi}{N} & & \cos\frac{(\frac{N}{4}-1)\pi}{N} & & & & \\
& & \iddots & & & & \ddots & & & \\
& \sin\frac{2\pi}{N} & & & & & & \cos\frac{2\pi}{N} & & \\
\sin\frac{\pi}{N} & & & & & & & & \cos\frac{\pi}{N} & 0 \\
0 & & & & 0 & & & & 0 & 1
\end{pmatrix}.$$

We note that the recursive sparse matrix factorization of DST-I matrix given by (4.47)–(4.49) is valid for $N = 4$ and 8.

4.4.3 The fast DCT-II/DST-II and DCT-III/DST-III algorithms

4.4.3.1 DCT-II recursive sparse matrix factorizations

The first direct real-valued fast algorithm for the DCT-II computation has been reported in Ref. [33] and it is based on the recursive sparse matrix factorization of DCT-II transform matrix defined as

$$C^{\mathrm{II}}_N = B_N \begin{pmatrix} \hat{C}^{\mathrm{II}}_{\frac{N}{2}} & 0 \\ 0 & \hat{C}^{\mathrm{IV}}_{\frac{N}{2}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.50}$$

where $\hat{C}^{\mathrm{II}}_{\frac{N}{2}}$ is the DCT-II matrix of half size with bit-reverse reordered rows, and $\hat{C}^{\mathrm{IV}}_{\frac{N}{2}} J_{\frac{N}{2}}$ is the DCT-IV matrix of half size with bit-reverse reordered rows and its columns in reverse order. B_N is a permutation matrix which permutes the transformed data sequence from the bit-reverse order to natural order. The recursive sparse matrix factorization (4.50) has become the fundamental form in the subsequent development of the direct real-valued fast DCT-II algorithms and it has initiated an extensive search to find an optimal factorization of the DCT-IV matrix [33, 36, 40, 41, 53, 55]. Essentially, the recursive sparse

matrix factorization (4.50) is evident when we consider the DCT-II in the form of a sum (the normalization factors are omitted) as follows:

$$c_k^{\mathrm{II}} = \sum_{n=0}^{N-1} x_n \cos \frac{\pi(2n+1)k}{2N}, \quad k = 0, 1, \ldots, N-1.$$

Splitting the sum into even-indexed and odd-indexed transform coefficients (i.e., grouping $2k$ and $2k+1$ terms) and using the symmetries of the transform kernels we get

$$c_{2k}^{\mathrm{II}} = \sum_{n=0}^{\frac{N}{2}-1} (x_n + x_{N-1-n}) \cos \frac{\pi(2n+1)k}{2(N/2)},$$

$$c_{2k+1}^{\mathrm{II}} = \sum_{n=0}^{\frac{N}{2}-1} (x_n - x_{N-1-n}) \cos \frac{\pi(2n+1)(2k+1)}{4(N/2)}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1.$$

Thus, the N-point DCT-II is recursively decomposed into an $\frac{N}{2}$-point DCT-II (even-indexed coefficients) and an $\frac{N}{2}$-point DCT-IV (odd-indexed coefficients). In general, since the factorization of DCT-IV matrix is the basis of direct real-valued fast DCT-II algorithms, all others proposed algorithms differ only by a modified/improved factorization of DCT-IV matrix which corresponds for a given N to the lower part in an appropriate signal flow graph. The generalized signal flow graph for the DCT-II computation with the proposed factorization of $C_{\frac{N}{2}}^{\mathrm{IV}}$ [33] for $N = 2$, 4 and 8 is shown in Fig. 4.8.

A slightly different recursive sparse matrix factorization of the DCT-II matrix and for completness of the DST-II matrix are respectively defined as [36, 40]

$$C_N^{\mathrm{II}} = P_N \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & J_{\frac{N}{2}} C_{\frac{N}{2}}^{\mathrm{IV}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.51}$$

$$S_N^{\mathrm{II}} = P_N \begin{pmatrix} S_{\frac{N}{2}}^{\mathrm{IV}} & 0 \\ 0 & J_{\frac{N}{2}} S_{\frac{N}{2}}^{\mathrm{II}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.52}$$

where P_N is a permutation matrix which reorders the transformed vector such that the first half are even-indexed coefficients in natural order, while the second half are odd-indexed coefficients but in reverse order. The generalized signal flow graph for the DCT-II computation with the proposed factorization of $C_{\frac{N}{2}}^{IV}$ [40] for $N = 2$, 4 and 8 is shown in Fig. 4.9. The fast DCT-II algorithm [40] with improved factorization of the DCT-IV matrix (see Section 4.4.4) in terms of reduced arithmetic complexity has been proposed in Ref. [41]. However, the proposed computational unit can be used for $N > 8$ (see Section 4.4.4.1).

The orthogonal recursive sparse matrix factorization of C_N^{II} matrix with scaling $\sqrt{2}$ has been introduced in Ref. [55]. For $N = 2^m$, $m > 1$, the DCT-II matrix C_N^{II} can be factorized

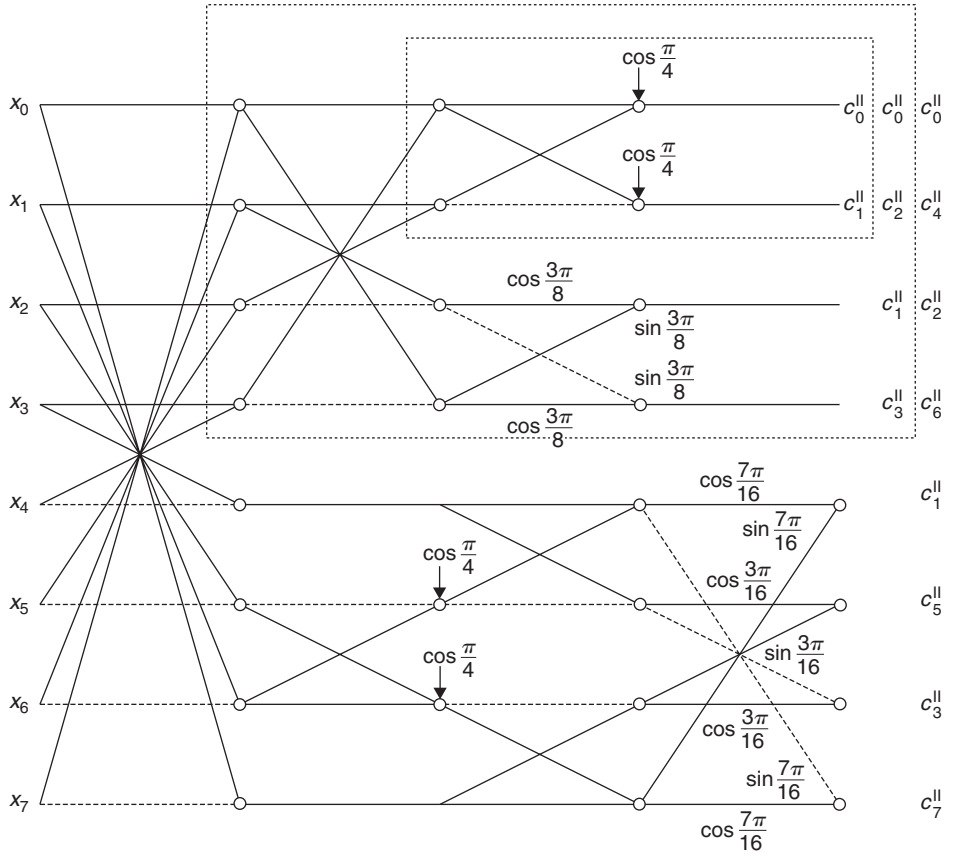


Fig. 4.8. The generalized signal flow graph for the DCT-II computation for $N = 2$, 4 and 8 based on (4.50).

in the form

$$C_N^{\mathrm{II}} = P_N^{\mathrm{T}} \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{IV}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ I_{\frac{N}{2}} & -J_{\frac{N}{2}} \end{pmatrix}$$

$$= P_N^{\mathrm{T}} \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{IV}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & J_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.53}$$

where P_N is a permutation matrix which when applied to a data vector corresponds to the reordering

$$\tilde{x}_n = x_{2n}, \quad \tilde{x}_{\frac{N}{2}+n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.54}$$

Note that $P_N^{\mathrm{T}} = P_N^{-1}$. By combining the orthogonal recursive sparse matrix factorizations of $C_{\frac{N}{2}}^{\mathrm{II}}$ and $C_{\frac{N}{2}}^{\mathrm{IV}}$ given by (4.53) and (4.91), respectively, we obtain the following factorization

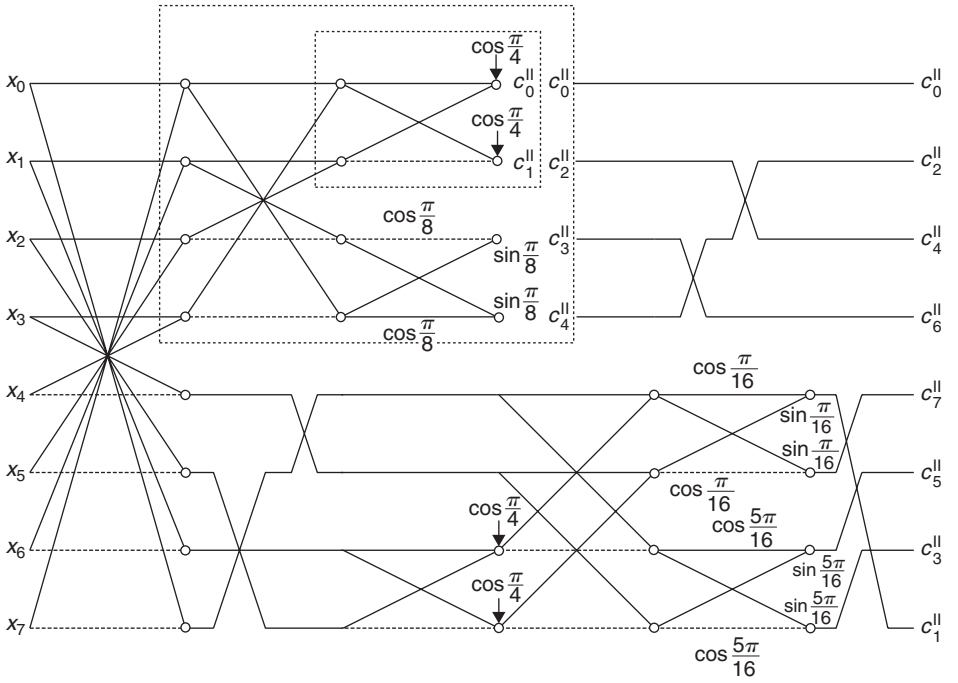


Fig. 4.9. The generalized signal flow graph for the DCT-II computation for $N = 2$, 4 and 8 based on (4.51).

of C_N^{II} matrix (scaled DCT-II) [55]:

$$C_N^{\mathrm{II}} = P_N^{\mathrm{T}} \begin{pmatrix} P_{\frac{N}{2}}^{\mathrm{T}} & 0 \\ 0 & P_{\frac{N}{2}}^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & A_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} C_{\frac{N}{4}}^{\mathrm{II}} & & & \\ & C_{\frac{N}{4}}^{\mathrm{IV}} & & \\ & & C_{\frac{N}{4}}^{\mathrm{II}} & \\ & & & C_{\frac{N}{4}}^{\mathrm{II}} \end{pmatrix}$$

$$\times \begin{pmatrix} T_{\frac{N}{2}}^{(0)} & 0 \\ 0 & T_{\frac{N}{2}}^{(1)} \end{pmatrix} T_N^{(0)}, \tag{4.55}$$

where $A_{\frac{N}{2}}$, $T_{\frac{N}{2}}^{(1)}$ and $T_{\frac{N}{2}}^{(0)}$, $T_N^{(0)}$ are orthogonal matrices defined as

$$A_{\frac{N}{2}} = \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & 0 \\ & I_{\frac{N}{4}-1} & I_{\frac{N}{4}-1} & \\ & I_{\frac{N}{4}-1} & -I_{\frac{N}{4}-1} & \\ 0 & & & -\sqrt{2} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{4}} & 0 \\ 0 & D_{\frac{N}{4}} J_{\frac{N}{4}} \end{pmatrix},$$

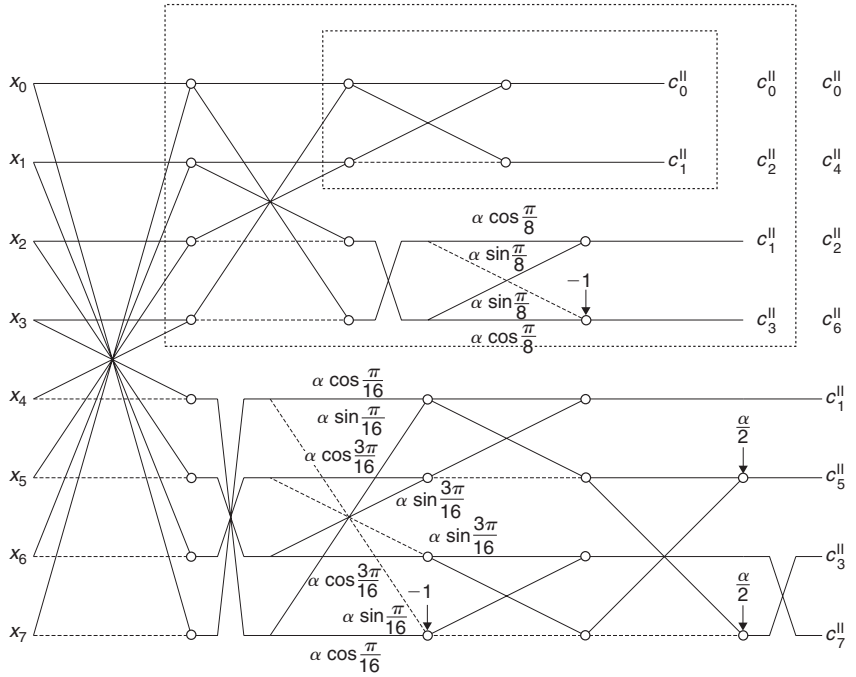


Fig. 4.10. The generalized signal flow graph for the DCT-II computation for $N = 2$, 4 and 8 based on (4.55); $\alpha = \sqrt{2}$.

$$T_{\frac{N}{2}}^{(1)} = \begin{pmatrix} I_{\frac{N}{4}} & 0 \\ 0 & D_{\frac{N}{4}} \end{pmatrix}$$

$$\times \begin{pmatrix} \cos\frac{\pi}{2N} & & & & & & \sin\frac{\pi}{2N} \\ & \cos\frac{2\pi}{2N} & & & & \sin\frac{2\pi}{2N} & \\ & & \ddots & & \iddots & & \\ & & & \cos\frac{(\frac{N}{4}-1)\pi}{2N} & \sin\frac{(\frac{N}{4}-1)\pi}{2N} & & \\ & & & -\sin\frac{(\frac{N}{4}-1)\pi}{2N} & \cos\frac{(\frac{N}{4}-1)\pi}{2N} & & \\ & & \iddots & & \ddots & & \\ & -\sin\frac{2\pi}{2N} & & & & \cos\frac{2\pi}{2N} & \\ -\sin\frac{\pi}{2N} & & & & & & \cos\frac{\pi}{2N} \end{pmatrix},$$

$$T_{\frac{N}{2}}^{(0)} = \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{4}} & J_{\frac{N}{4}} \\ I_{\frac{N}{4}} & -J_{\frac{N}{4}} \end{pmatrix} = \begin{pmatrix} I_{\frac{N}{4}} & 0 \\ 0 & J_{\frac{N}{4}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{4}} & J_{\frac{N}{4}} \\ J_{\frac{N}{4}} & -I_{\frac{N}{4}} \end{pmatrix},$$

where $D_{\frac{N}{4}} = \text{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{4} - 1$ is the diagonal odd-sign changing matrix. The corresponding generalized signal flow graph for the DCT-II computation for $N = 2$, 4 and 8 is shown in Fig. 4.10. For $N = 8$, each output coefficient should be normalized by scaling factor $\frac{\sqrt{2}}{4}$ to get the true DCT-II coefficients.

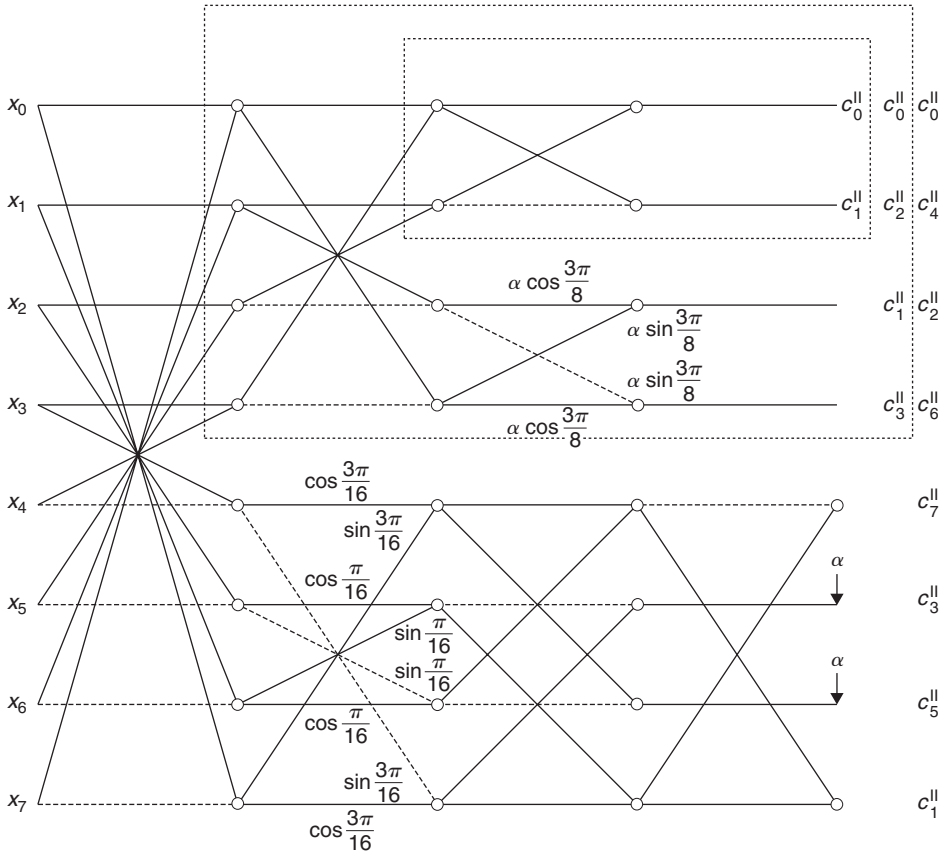


Fig. 4.11. The signal flow graph for the scaled DCT-II computation for $N = 8$; $\alpha = \sqrt{2}$. © 1989, IEEE.

Finally, it is important to present a practical fast algorithm from the class of fast 8-point DCT-II algorithms generated from the full matrix equation in a systematic way using graph transformations and equivalence relations [45]. In the definition of DCT-II the scaling constant $\sqrt{2}$ has been introduced which resulted in $\sqrt{2}\epsilon_k = 1$ for $k = 0$ (scaled DCT-II) allowing for the coefficient c_0^{II} to be evaluated without any multiplication. The 8-point DCT-II computation requires 11 multiplications and 29 additions, thus achieving the theoretical lower bound of the number of multiplications for $N = 8$. The corresponding signal flow graph for scaled DCT-II computation for $N = 8$ is shown in Fig. 4.11. For $N = 8$, each output coefficient should be normalized by scaling factor $\frac{1}{\sqrt{8}}$ to get the true DCT-II coefficients. We note that the algorithm defined by (4.55) achieves also the theoretical lower bound of the number of multiplications for $N = 8$.

In Section 4.4.1.3, we observe from the split-radix fast DCT-I algorithm that there exists a factorization of N-point DCT-III matrix C_N^{III} (and hence S_N^{III} too) based on $(\frac{N}{2} - 1)$-point DST-I and $(\frac{N}{2} + 1)$-point DCT-I. Actually, such orthogonal recursive sparse matrix factorizations of C_N^{III} and S_N^{III} transform matrices with scaling $\sqrt{2}$ are respectively

defined as [55]

$$C_N^{\mathrm{III}} = P_N^{\mathrm{T}} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & J_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & \\ & I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & & \sqrt{2} & \\ & J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & (-1)^{\frac{N}{2}} D_{\frac{N}{2}-1} \end{pmatrix}$$
$$\times \begin{pmatrix} C^{\mathrm{I}}_{\frac{N}{2}+1} & 0 \\ 0 & S^{\mathrm{I}}_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & D_{\frac{N}{2}-1} \end{pmatrix} R_N, \tag{4.56}$$

and

$$S_N^{\mathrm{III}} = P_N^{\mathrm{T}} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & -I_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & \\ & I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & & \sqrt{2} & \\ & J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & (-1)^{\frac{N}{2}} D_{\frac{N}{2}-1} \end{pmatrix}$$
$$\times \begin{pmatrix} C^{\mathrm{I}}{}_{\frac{N}{2}+1} & 0 \\ 0 & S^{\mathrm{I}}{}_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}+1} & 0 \\ 0 & D_{\frac{N}{2}-1} \end{pmatrix} R_N \, J_N, \tag{4.57}$$

where $D_{\frac{N}{2}-1} = \mathrm{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{2} - 2$, is the diagonal odd-sign changing matrix, P_N is a permutation matrix defined by (4.54) and R_N is the rotation matrix given by

$$R_N = \begin{pmatrix} 1 & 0 & & & & & & 0 \\ 0 & \cos\frac{\pi}{2N} & & & & & & \sin\frac{\pi}{2N} \\ & & \cos\frac{2\pi}{2N} & & & & \sin\frac{2\pi}{2N} & \\ & & & \ddots & & \iddots & & \\ & & & & \cos\frac{(\frac{N}{2}-1)\pi}{2N} & \sin\frac{(\frac{N}{2}-1)\pi}{2N} & & \\ & & & & -\sin\frac{(\frac{N}{2}-1)\pi}{2N} & \cos\frac{(\frac{N}{2}-1)\pi}{2N} & & \\ & & & \iddots & & \ddots & & \\ & & -\sin\frac{2\pi}{2N} & & & & \cos\frac{2\pi}{2N} & \\ 0 & -\sin\frac{\pi}{2N} & & & & & & \cos\frac{\pi}{2N} \end{pmatrix}.$$

4.4.3.2 DCT-II computation via Walsh–Hadamard transform

Since any EOT (see Section 4.3) can be expressed in terms of any other EOT through a conversion matrix [34], the DCT-II with even symmetry/even antisymmetry structure of basis vectors can be realized via other simpler EOT [4, 28, 34, 37] such as sequency

(Walsh) ordered Walsh–Hadamard transform (WHT) [1] whose basis vectors consist of ± 1 elements only.

Denote the DCT-II and WHT respectively in the matrix-vector notation (the normalization factors are omitted) as

$$\mathbf{c}^{\mathrm{II}} = C_N^{\mathrm{II}}\mathbf{x}^{\mathrm{T}}, \quad \mathbf{w} = W_N\mathbf{x}^{\mathrm{T}}. \tag{4.58}$$

Examination of C_N^{II} and W_N matrices for a given N shows that there is a one-to-one correspondence between the sequencies (the number of zero-crossings in sign) of the rows of these transform matrices. This implies that the even symmetry/even antisymmetry structure of the WHT matrix is preserved in the DCT-II matrix. If we rearrange the rows of C_N^{II} and W_N in bit-reversed order, then (4.58) can be rewritten as

$$\hat{\mathbf{c}}^{\mathrm{II}} = \hat{C}_N^{\mathrm{II}}\mathbf{x}^{\mathrm{T}}, \quad \hat{\mathbf{w}} = \hat{W}_N\mathbf{x}^{\mathrm{T}}. \tag{4.59}$$

Since $\hat{W}_N$ is an orthonormal matrix, substituting the relation $\hat{W}_N^{\mathrm{T}}\hat{W}_N = I_N$ into the (4.59) we get [34]

$$\hat{\mathbf{c}}^{\mathrm{II}} = \hat{C}_N^{\mathrm{II}}\hat{W}_N^{\mathrm{T}}\hat{W}_N\mathbf{x}^{\mathrm{T}} = T_N\hat{\mathbf{w}}^{\mathrm{T}}, \quad T_N = \hat{C}_N^{\mathrm{II}}\hat{W}_N^{\mathrm{T}}, \tag{4.60}$$

where T_N is the conversion matrix which takes the Walsh domain vector and converts it to the DCT-II domain. The conversion matrix T_N has two important properties:

1. It is orthonormal, being the product of two orthonormal matrices $\hat{C}_N^{\mathrm{II}}$ and $\hat{W}_N^{\mathrm{T}}$.
2. It has a sparse block diagonal structure as long as the rows of C_N^{II} and W_N are in bit-reversed order.

Equation (4.60) defines the fast algorithm for DCT-II computation via WHT involving two steps: the efficient WHT computation and the implementation of the conversion matrix T_N. In order to illustrate the DCT-II computation via WHT consider $N = 8$. The W_8 matrix is given by [1]

$$W_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix},$$

and the conversion matrix $T_8 = \hat{C}_8^{\mathrm{II}}\hat{W}_8^{\mathrm{T}}$ in analytical form is given by

$$T_8 = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & U_2 & \\ 0 & & & U_4 \end{pmatrix},$$

where

$$U_2 = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix},$$

$$U_4 = \begin{pmatrix} \cos\frac{\pi}{8}\cos\frac{\pi}{16} & -\sin\frac{\pi}{8}\sin\frac{\pi}{16} & \sin\frac{\pi}{8}\cos\frac{\pi}{16} & \cos\frac{\pi}{8}\sin\frac{\pi}{16} \\ \sin\frac{\pi}{8}\sin\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{3\pi}{16} & -\cos\frac{\pi}{8}\sin\frac{3\pi}{16} & \sin\frac{\pi}{8}\cos\frac{3\pi}{16} \\ -\sin\frac{\pi}{8}\cos\frac{3\pi}{16} & \cos\frac{\pi}{8}\sin\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{3\pi}{16} & \sin\frac{\pi}{8}\sin\frac{3\pi}{16} \\ -\cos\frac{\pi}{8}\sin\frac{\pi}{16} & -\sin\frac{\pi}{8}\cos\frac{\pi}{16} & -\sin\frac{\pi}{8}\sin\frac{\pi}{16} & \cos\frac{\pi}{8}\cos\frac{\pi}{16} \end{pmatrix}.$$

The $M \times M$ block matrices U_M in the conversion matrix T_N possessing the following general structure [38]

$$U_M = \begin{pmatrix} A^{(1)}_{\frac{M}{2}} & A^{(2)}_{\frac{M}{2}} \\ -J_{\frac{M}{2}} A^{(2)}_{\frac{M}{2}} J_{\frac{M}{2}} & J_{\frac{M}{2}} A^{(1)}_{\frac{M}{2}} J_{\frac{M}{2}} \end{pmatrix}, \quad M = 2, 4, 8, \ldots, \frac{N}{2}, \tag{4.61}$$

can be further factorized into a product of sparse matrices. The higher-order conversion matrix T_{2N} can be generated recursively [37], i.e., having derived the block matrices $U_2, U_4, \ldots, U_{\frac{N}{2}}$ for T_N we need to derive for the T_{2N} only the block matrix U_N taking into account its above general structure.

Among the existing sparse factorizations of U_4 matrix [35, 37], the factorization presented in Ref. [37] is preferred in terms of structural simplicity and regularity and it is defined as

$$U_4 = P_4 \begin{pmatrix} \cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & 0 & 0 \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & 0 & 0 \\ 0 & 0 & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ 0 & 0 & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} P_4,$$

where

$$P_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

For the fast WHT computation, Manz's algorithm is presented in Ref. [1] requiring the input data sequence $\{x_n\}$ to be in bit-reversed order. Since WHT is an EOT according to

(4.17) we can obtain more suitable factorization of $\hat{W}_8$ matrix as

$$\hat{W}_8 = \begin{pmatrix} 1 & 1 & 1 & 1 & & & & \\ 1 & -1 & -1 & 1 & & 0 & & \\ 1 & 1 & -1 & -1 & & & & \\ 1 & -1 & 1 & -1 & & & & \\ & & & & 1 & 1 & 1 & 1 \\ & & & & 1 & -1 & -1 & 1 \\ & & 0 & & -1 & -1 & 1 & 1 \\ & & & & -1 & 1 & -1 & 1 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix},$$

where

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix},$$

and

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ -1 & -1 & 1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ -J_2 & I_2 \end{pmatrix},$$

defining the new alternative fast WHT algorithm for $N = 8$. The corresponding signal flow graph for the DCT-II computation via WHT for $N = 8$ including $N = 4$ is shown in Fig. 4.12.

If we compare the signal flow graph for $N = 8$ in Fig. 4.12 with the DCT-II implementation via WHT presented in Ref. [35] the implementation here has the following advantages:

- Saves two bit-reversal permutations.
- For 5 plane rotations the number of rotation angles is reduced from four to three.
- The implementation is more regular.

4.4.3.3 DCT-II computation via DFT of real-valued data

A fast algorithm with simple recursive structure for the efficient evaluation both of the DFT and DCT-II, called the fast Fourier-cosine transform (FFCT), has been proposed in Refs. [42, 43]. Principally both problems, the efficient evaluation of the DFT and DCT-II, are closely related since a DCT-II of dimension N can be mapped into a DFT of the same size and output plane rotations, and since a DFT of dimension N can be mapped into DFT of length $\frac{N}{2}$ and two DCTs-II of length $\frac{N}{4}$. The method is recursively applied again, until trivial only transforms remain.

Before presenting the complete formulae of FFCT algorithm let us define the following discrete transforms of a real-valued data vector $\mathbf{x}$ of length $N = 2^m$ (the normalization

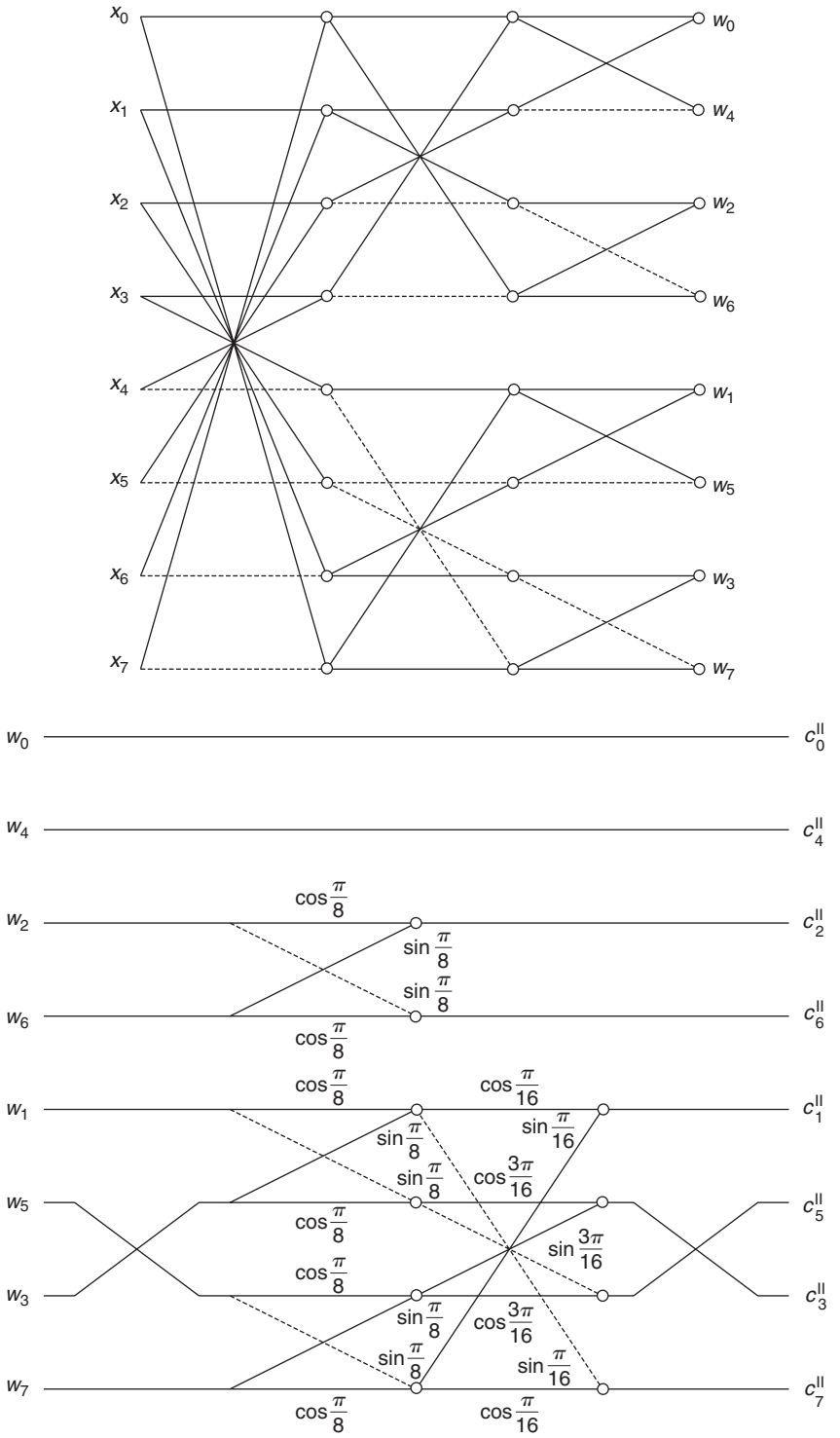


Fig. 4.12. The signal flow graph for the DCT-II computation via WHT for $N = 8$ (the fast WHT and conversion matrix).

factors are omitted) using the original notation introduced in Ref. [42]:

$$\mathrm{DFT}(k,N,\mathbf{x}) = \sum_{n=0}^{N-1} x_n \mathrm{e}^{-\mathrm{j}\frac{2\pi nk}{N}}, \quad k = 0,1,\ldots,N-1, \quad \mathrm{j} = \sqrt{-1}, \tag{4.62}$$

$$\mathrm{DCT}(k,N,\mathbf{x}) = \sum_{n=0}^{N-1} x_n \cos\frac{2\pi(2n+1)k}{4N}, \quad k = 0,1,\ldots,N-1, \tag{4.63}$$

$$cos-\mathrm{DFT}(k,N,\mathbf{x}) = \sum_{n=0}^{N-1} x_n \cos\frac{2\pi nk}{N}, \quad k = 0,1,\ldots,N-1, \tag{4.64}$$

$$sin-\mathrm{DFT}(k,N,\mathbf{x}) = \sum_{n=0}^{N-1} x_n \sin\frac{2\pi nk}{N}, \quad k = 0,1,\ldots,N-1. \tag{4.65}$$

It is simple to verify that the following relations hold:

$$\mathrm{DFT}(k,N,\mathbf{x}) = cos-\mathrm{DFT}(k,N,\mathbf{x}) - i\,sin-\mathrm{DFT}(k,N,\mathbf{x}), \tag{4.66}$$

$$\mathrm{DCT}(N,N,\mathbf{x}) = 0, \tag{4.67}$$

$$\mathrm{DCT}(-k,N,\mathbf{x}) = \mathrm{DCT}(k,N,\mathbf{x}), \tag{4.68}$$

$$\mathrm{DCT}(2N-k,N,\mathbf{x}) = -\mathrm{DCT}(k,N,\mathbf{x}), \tag{4.69}$$

$$cos-\mathrm{DFT}(N-k,N,\mathbf{x}) = cos-\mathrm{DFT}(k,N,\mathbf{x}), \tag{4.70}$$

$$sin-\mathrm{DFT}(N-k,N,\mathbf{x}) = -sin-\mathrm{DFT}(k,N,\mathbf{x}). \tag{4.71}$$

The complete formulae of the FFCT algorithm for the efficient evaluation of the DCT-II are given by

$$\begin{pmatrix} \mathrm{DCT}(k,N,\mathbf{x}) \\ \mathrm{DCT}(N-k,N,\mathbf{x}) \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi k}{2N} & -\sin\frac{\pi k}{2N} \\ \sin\frac{\pi k}{2N} & \cos\frac{\pi k}{2N} \end{pmatrix} \begin{pmatrix} cos-DFT(k,N,\tilde{\mathbf{x}}) \\ sin-DFT(k,N,\tilde{\mathbf{x}}) \end{pmatrix},$$

$$k = 0,1,\ldots,\frac{N}{2}-1,$$

$$\mathrm{DCT}(N/2,N,\mathbf{x}) = \frac{\sqrt{2}}{2}\, cos-\mathrm{DFT}(N/2,N,\tilde{\mathbf{x}}), \tag{4.72}$$

where

$$\tilde{x}_n = x_{2n}, \;\; \tilde{x}_{N-1-n} = x_{2n+1}, \quad n = 0,1,\ldots,\frac{N}{2}-1. \tag{4.73}$$

Hence, the N-point DCT-II is mapped into the DFT of the same size and output plane rotations. The transforms $cos-\mathrm{DFT}(k,N,\tilde{\mathbf{x}})$ and $sin-\mathrm{DFT}(k,N,\tilde{\mathbf{x}})$ are respectively

evaluated as

$$cos - \mathrm{DFT}(k, N, \tilde{\mathbf{x}}) = cos - \mathrm{DFT}(k, N/2, \tilde{\mathbf{x}}^{(1)}) + \mathrm{DCT}(k, N/4, \tilde{\mathbf{x}}^{(2)}),$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.74}$$

with

$$\tilde{x}_n^{(1)} = \tilde{x}_{2n}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.75}$$

$$\tilde{x}_n^{(2)} = \tilde{x}_{2n+1} + \tilde{x}_{N-1-2n}, \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.76}$$

and

$$sin - \mathrm{DFT}(k, N, \tilde{\mathbf{x}}) = sin - \mathrm{DFT}(k, N/2, \tilde{\mathbf{x}}^{(1)}) + \mathrm{DCT}\left(\frac{N}{4} - k, N/4, \tilde{\mathbf{x}}^{(3)}\right),$$
$$k = 0, 1, \ldots, \frac{N}{2} - 1, \tag{4.77}$$

with

$$\tilde{x}_n^{(3)} = (-1)^n(\tilde{x}_{2n+1} - \tilde{x}_{N-1-2n}), \quad n = 0, 1, \ldots, \frac{N}{4} - 1, \tag{4.78}$$

whereby (4.69), (4.70) and (4.71) are applied when necessary. The transform $cos - \mathrm{DFT}$ of dimension N is decomposed into $cos - \mathrm{DFT}$ of length $\frac{N}{2}$ and the DCT-II of length $\frac{N}{4}$. Similarly, the transform $sin - \mathrm{DFT}$ of dimension N is decomposed into $sin - \mathrm{DFT}$ of length $\frac{N}{2}$ and the DCT-II of length $\frac{N}{4}$. The transforms are recursively reduced to lower sizes until they become trivial. The FFCT algorithm for $N = 8$ results in the signal flow graph shown in Fig. 4.13.

4.4.3.4 The split-radix DCT-II algorithm

Besides the DCT-I and DST-I the idea of 2^m split-radix FFT algorithm [24–26] can also be extended to the DCT-II. The complete formulae of split-radix fast DCT-II algorithm (normalization factors are omitted) are given by

$$c_{2k}^{\mathrm{II}} = \sum_{n=0}^{\frac{N}{2}} (x_n + x_{N-1-n}) \cos\frac{\pi(2n+1)k}{2(N/2)}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1,$$
$$c_{4k-1}^{\mathrm{II}} = a_k + b_k, \quad k = 1, 2, \ldots, \frac{N}{4},$$
$$c_{4k+1}^{\mathrm{II}} = a_k - b_k, \quad k = 0, 1, \ldots, \frac{N}{4} - 1, \ \ b_0 = 0, \tag{4.79}$$

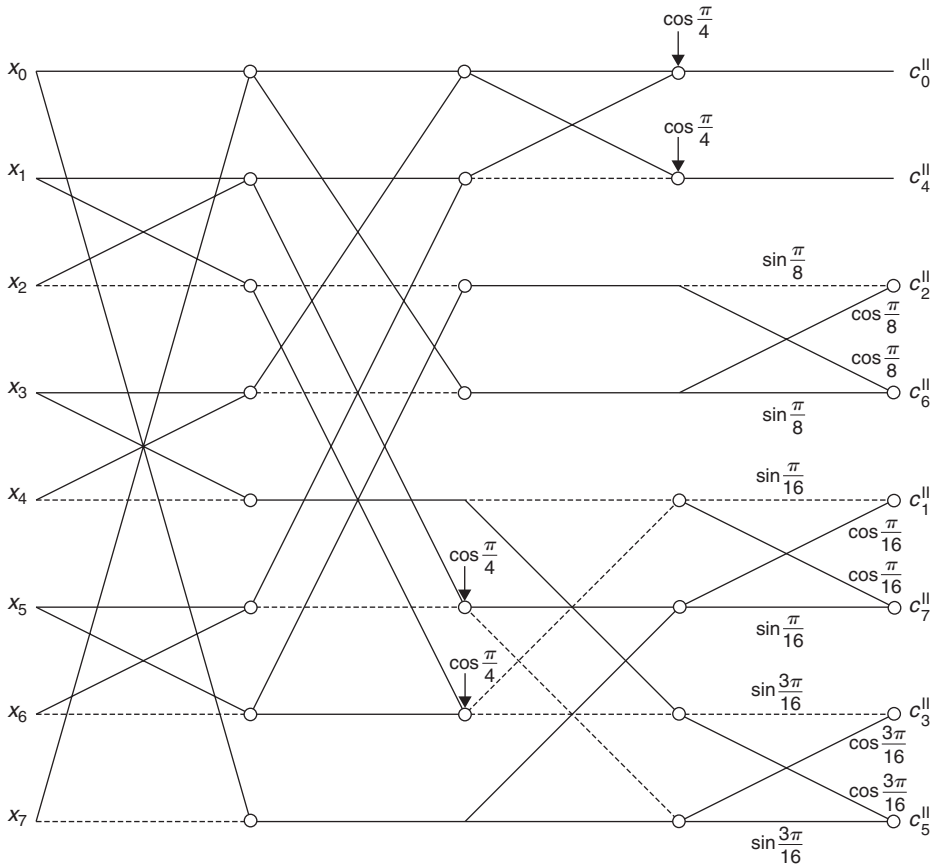


Fig. 4.13. The signal flow graph for the DCT-II computation via DFT of real-valued data for $N = 8$.

where

$$a_k = \sum_{n=0}^{\frac{N}{4}-1} \left[(x_n - x_{N-1-n}) \cos \frac{\pi(2n+1)}{2N} + (x_{\frac{N}{2}-1-n} - x_{\frac{N}{2}+n}) \sin \frac{\pi(2n+1)}{2N} \right] \times \cos \frac{\pi(2n+1)k}{2(N/4)},$$

$$b_k = \sum_{n=1}^{\frac{N}{4}-1} \left[(x_n - x_{N-1-n}) \sin \frac{\pi(2n+1)}{2N} - (x_{\frac{N}{2}-1-n} - x_{\frac{N}{2}+n}) \cos \frac{\pi(2n+1)}{2N} \right] \times \sin \frac{\pi(2n+1)k}{2(N/4)}. \tag{4.80}$$

The first stage of split-radix decomposition replaces N-point DCT-II by one $\frac{N}{2}$-point DCT-II, one DCT-II of length $\frac{N}{4}$ and one DST-II of length $\frac{N}{4}$. The decomposition is used recursively. It defines the recursive sparse matrix factorization of the DCT-II matrix given by

$$C_N^{\mathrm{II}} = B_N \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & K_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}, \tag{4.81}$$

where B_N is a permutation matrix which permutes the transformed data sequence from bit-reverse order to natural order. The matrix $K_{\frac{N}{2}}$ is given by

$$K_{\frac{N}{2}} = \begin{pmatrix} 1 & & & & & 0 \\ & \cos\frac{\pi}{4} & & & -\sin\frac{\pi}{4} & \\ & & I_{\frac{N}{4}-2} & -I_{\frac{N}{4}-2} & & \\ & & I_{\frac{N}{4}-2} & I_{\frac{N}{4}-2} & & \\ & \sin\frac{\pi}{4} & & & \cos\frac{\pi}{4} & \\ 0 & & & & & 1 \end{pmatrix} \begin{pmatrix} C_{\frac{N}{4}}^{\mathrm{II}} & 0 \\ 0 & S_{\frac{N}{4}}^{\mathrm{II}} J_{\frac{N}{4}} \end{pmatrix} R_{\frac{N}{2}}, \tag{4.82}$$

where $R_{\frac{N}{2}}$ is a rotation matrix given by

$$R_{\frac{N}{2}} = \begin{pmatrix} \sin\frac{\pi}{2N} & & & & & & \cos\frac{\pi}{2N} \\ & \sin\frac{3\pi}{2N} & & & & \cos\frac{3\pi}{2N} & \\ & & \ddots & & \iddots & & \\ & & & \sin\frac{(\frac{N}{2}-1)\pi}{2N} & \cos\frac{(\frac{N}{2}-1)\pi}{2N} & & \\ & & & -\cos\frac{(\frac{N}{2}-1)\pi}{2N} & \sin\frac{(\frac{N}{2}-1)\pi}{2N} & & \\ & & \iddots & & \ddots & & \\ & -\cos\frac{3\pi}{2N} & & & & \sin\frac{3\pi}{2N} & \\ -\cos\frac{\pi}{2N} & & & & & & \sin\frac{\pi}{2N} \end{pmatrix}.$$

The corresponding generalized signal flow graph for the DCT-II computation based on split-radix algorithm for $N = 2$, 4 and 8 is shown in Fig. 4.14. The output data sequence $\{c_k^{\mathrm{II}}\}$ is in bit-reversed order.

The recursive sparse matrix factorization given by (4.81) and (4.82) indicates two interesting facts. First, comparing with the factorization (4.50) it can be seen that the matrix $K_{\frac{N}{2}}$ represents a new factorization of $C_{\frac{N}{2}}^{\mathrm{IV}}$ transform matrix consisting of a rotation matrix, $\frac{N}{4}$-point DCT-II and $\frac{N}{4}$-point DST-II, whose outputs are combined in the final stage. Such a factorization is actually defined by (4.91) in Section 4.4.4.1. Secondly, from the generalized signal flow graph shown in Fig. 4.14 it is clear that using the definition of scaled

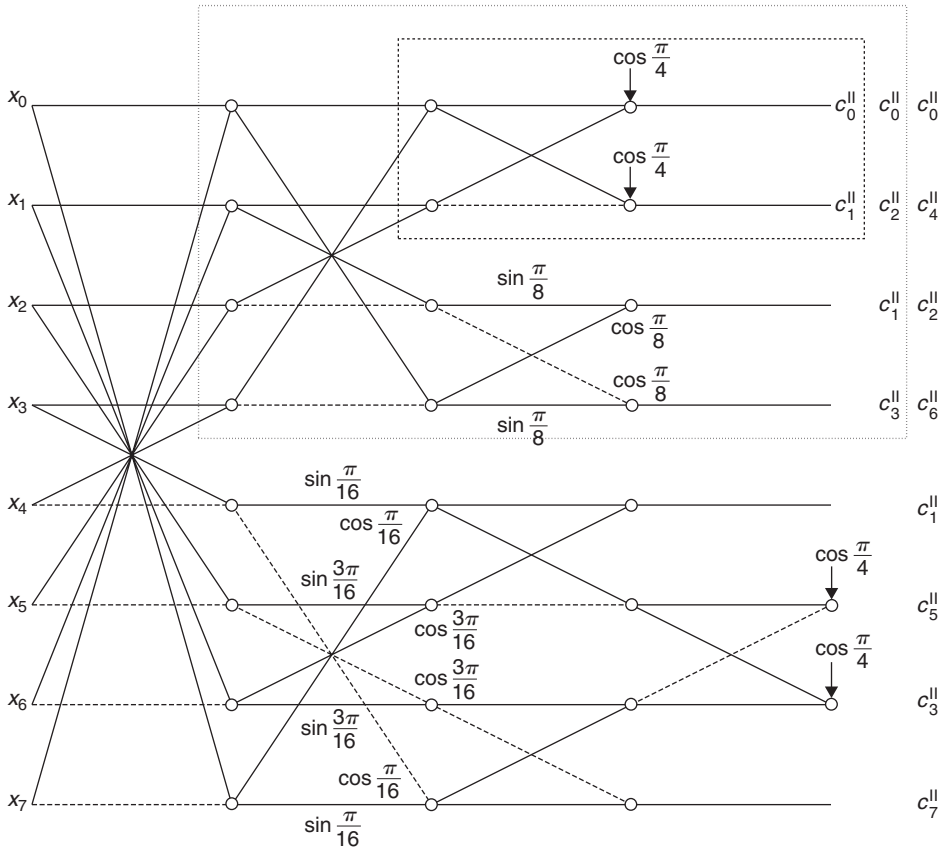


Fig. 4.14. The generalized signal flow graph for the DCT-II computation for $N = 2$, 4 and 8 based on the split-radix algorithm.

DCT-II, the split-radix algorithm for $N = 8$ achieves also the theoretical lower bound of the number of multiplications.

Note: The DCT-III/DST-III sparse matrix factorizations are simply obtained by transposition of DCT-II/DST-II ones and the fast DCT-III/DST-III algorithms are obtained from DCT-II/DST-II ones but performed in opposite direction.

4.4.4 The fast DCT-IV/DST-IV algorithms

In general, the efficient computation of DCT-IV/DST-IV can be realized in two ways:

1. Because of the relationship between DCT-II and DCT-IV matrices given by (4.13), the DCT-IV of size N may be converted to the DCT-II of the same size at the cost of additional N pre-multiplications and $N - 1$ recursive post-additions [48, 49]. Thus, the existing fast DCT-II algorithms can be directly used for the efficient DCT-IV computation.

2. The DCT-II of size N is decomposed into $\frac{N}{2}$-point DCT-II and $\frac{N}{2}$-point DCT-IV (see (4.50)). This fact leads to the following accepted conclusion [51]: the fast N-point DCT-IV algorithms can be indirectly derived from existing fast $2N$-point DCT-II algorithms.

4.4.4.1 DCT-IV recursive sparse matrix factorizations

The first attempt to derive a direct DCT-IV sparse matrix factorization was presented in Ref. [33], which at that time was not very efficient. This result was reconsidered [36] and a factorization of DCT-IV matrix consisting of simple sparse buttefly binary and cosine/sine matrices [40] was proposed. Subsequently, the sparse matrix factorization [40] was improved identifying its more efficient basic computational unit [41].

The most compact form of the direct sparse matrix factorization of DCT-IV matrix is formulated in Ref. [41]. For $N = 2^m$, the DCT-IV matrix can be factorized into the following product of sparse matrices:

$$C_N^{\mathrm{IV}} = P_N V_N^{(m)} Y_N^{(1)} Y_N^{(2)} \dots Y_N^{(m-2)} Y_N^{(m-1)} H_N, \tag{4.83}$$

where P_N is a permutation matrix that reverses the order of odd-indexed components of a data vector. $V_N^{(m)}$ is a cosine/sine block diagonal matrix formed by

$$V_N^{(m)} = \mathrm{diag}\{T_{\frac{1}{4N}}, T_{\frac{5}{4N}}, \dots, T_{\frac{2N-3}{4N}}\}, \quad T_r = \begin{pmatrix} \cos r\pi & \sin r\pi \\ \sin r\pi & -\cos r\pi \end{pmatrix},$$

$Y_N^{(1)}$ is a matrix consisting of the product of following matrices

$$Y_N^{(1)} = R_N^{(m-1)} X_N^{(m-1)} R_N^{(m-2)} X_N^{(m-2)} \dots R_N^{(1)} X_N^{(1)},$$

where $R_N^{(i)}$ and $X_N^{(i)}$, $i = 1, 2, \dots, m-1$ are respectively binary and cosine/sine block diagonal matrices defined as

$$R_N^{(i)} = \mathrm{diag}\{I_{N-2^{i+1}}, U_{2^{i+1}}^{(i)}\}, \quad U_M^{(i)} = \frac{1}{\sqrt{2}}\mathrm{diag}\{B^{(i)}, B^{(i)}, \dots, B^{(i)}\}, \quad B^{(i)} = \begin{pmatrix} I_{2^i} & I_{2^i} \\ I_{2^i} & -I_{2^i} \end{pmatrix}$$

and

$$X_N^{(i)} = \mathrm{diag}\{I_{N-2^i}, E^{(i)}\}, \quad E^{(i)} = \mathrm{diag}\{T_{\frac{1}{2^{i+1}}}, T_{\frac{5}{2^{i+1}}}, \dots, T_{\frac{2^{i+1}-3}{2^{i+1}}}\}.$$

All remaining matrices $Y_N^{(2)}, Y_N^{(3)} \dots, Y_N^{(m-1)}$ are defined as

$$\begin{aligned} Y_N^{(2)} &= \mathrm{diag}\{Y_{\frac{N}{2}}^{(1)}, I_{\frac{N}{2}}\}, \\ Y_N^{(3)} &= \mathrm{diag}\{Y_{\frac{N}{4}}^{(1)}, I_{\frac{N}{4}}, Y_{\frac{N}{4}}^{(1)}, I_{\frac{N}{4}}\}, \\ &\vdots \qquad\qquad \vdots \\ Y_N^{(m-1)} &= \mathrm{diag}\{Y_4^{(1)}, I_4, \dots, Y_4^{(1)}, I_4\}. \end{aligned}$$

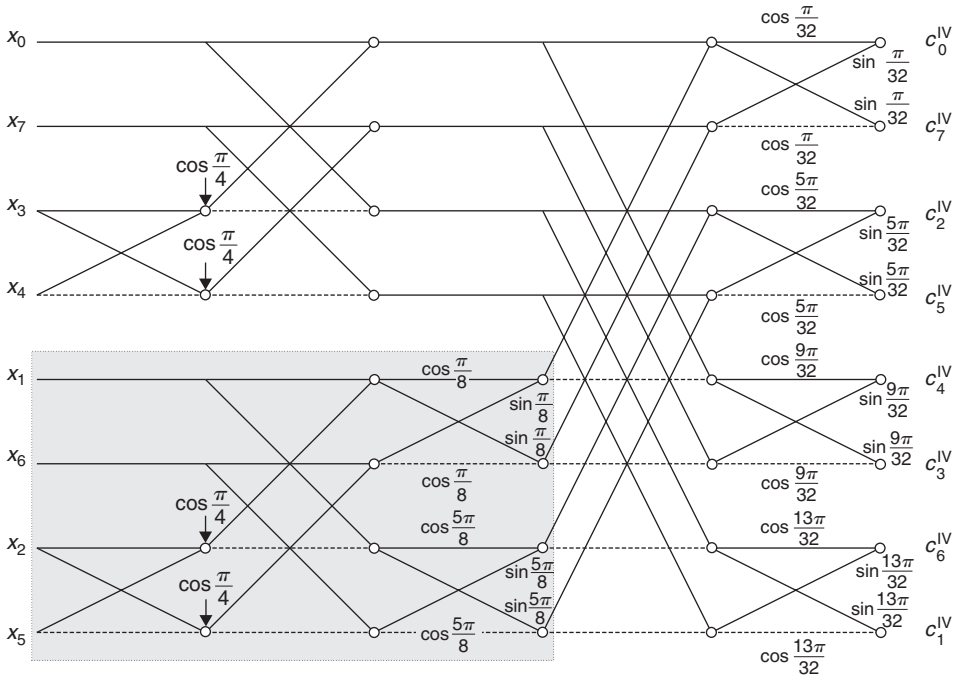


Fig. 4.15. The signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ based on (4.83) and (4.85).

Finally, the last matrix H_N in (4.83) is also a permutation matrix that changes an increasing index into a Hadamard index. To be more specific, let $h_N(i)$ represent the sequency (the number of sign changes) of the i-th row of $N \times N$ Hadamard matrix [1]. If a data sequence is ordered according to $h_N(i)$ then we say that the data sequence is in the Hadamard order. The $h_N(i)$ can be recursively generated as [50]

$$h_{2N}(2i) = h_N(i), \;\; h_{2N}(2i+1) = 2N - 1 - h_N(i), \quad i = 1, 2, \ldots, N-1 \tag{4.84}$$

with initial conditions $h_1(0) = 0$ and $h_2(1) = 1$. The corresponding signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ is shown in Fig. 4.15.

The improved factorization of C_N^{IV} matrix [41] is based on replacing the computational unit

$$Z(k,l) = \begin{pmatrix} \cos\frac{l}{2k} & \sin\frac{l}{2k} & 0 & 0 \\ \sin\frac{l}{2k} & -\cos\frac{l}{2k} & 0 & 0 \\ 0 & 0 & \cos\frac{l+k}{2k} & \sin\frac{l+k}{2k} \\ 0 & 0 & \sin\frac{l+k}{2k} & -\cos\frac{l+k}{2k} \end{pmatrix} \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} \begin{pmatrix} I_2 & 0 & 0 \\ 0 & \cos\frac{l}{k} & \sin\frac{l}{k} \\ 0 & \sin\frac{l}{2} & -\cos\frac{l}{k} \end{pmatrix} \tag{4.85}$$

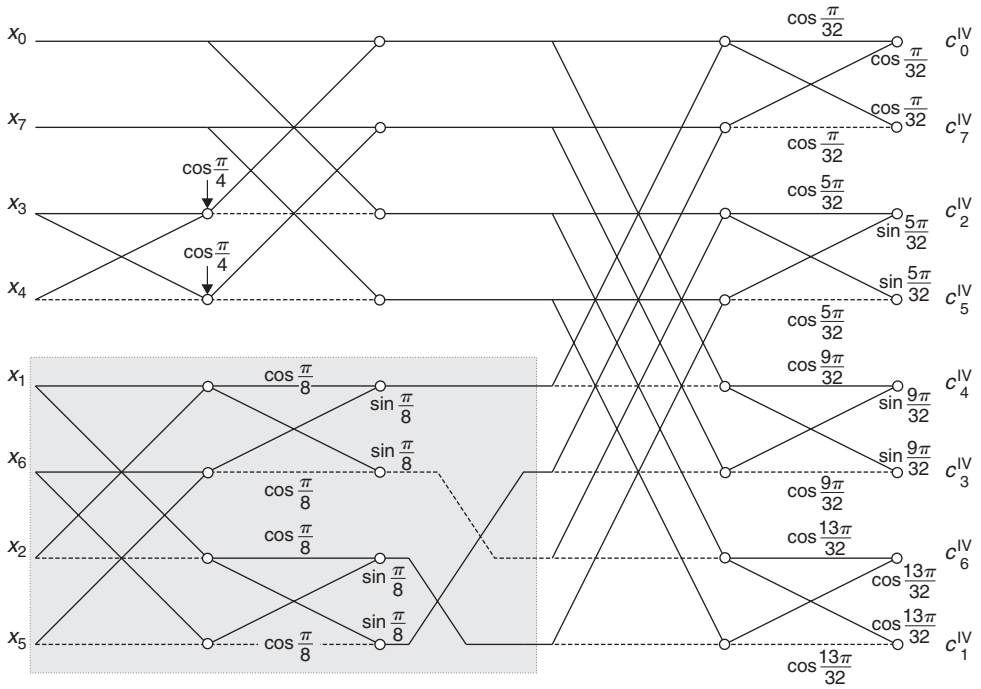


Fig. 4.16. The signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ with improved factorization of DCT-IV matrix (4.83) and (4.86).

by a more efficient one defined as [41]

$$Z(k,l) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} \cos\frac{l}{2k} & \sin\frac{l}{2k} & 0 & 0 \\ \sin\frac{l}{2k} & -\cos\frac{l}{2k} & 0 & 0 \\ 0 & 0 & \cos\frac{l}{2k} & \sin\frac{l}{2k} \\ 0 & 0 & \sin\frac{l}{2k} & -\cos\frac{l}{2k} \end{pmatrix} \begin{pmatrix} I_2 & I_2 \\ I_2 & -I_2 \end{pmatrix} \quad (4.86)$$

The corresponding signal flow graph for $N = 8$ is shown in Fig. 4.16. One can compare the highlighted basic computational units in Figs. 4.15 and 4.16.

A slightly different sparse matrix factorization of the DCT-IV matrix for $N = 2^m$ is defined as [32, 47]

$$C_N^{\mathrm{IV}} = R_N \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{III}} & 0 \\ 0 & J_{\frac{N}{2}} S_{\frac{N}{2}}^{\mathrm{III}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \sqrt{2} & & & \\ & I_{\frac{N}{2}-1} & & J_{\frac{N}{2}-1} \\ & & \sqrt{2} & \\ & -J_{\frac{N}{2}-1} & & I_{\frac{N}{2}-1} \end{pmatrix} P_N, \quad (4.87)$$

where R_N is the rotation matrix (Householder reflections) given by

$$R_N = \begin{pmatrix} \cos\frac{\pi}{4N} & & & & & & \sin\frac{\pi}{4N} \\ & \cos\frac{3\pi}{4N} & & & & \sin\frac{3\pi}{4N} & \\ & & \ddots & & \iddots & & \\ & & & \cos\frac{(N-1)\pi}{4N} & \sin\frac{(N-1)\pi}{4N} & & \\ & & & \sin\frac{(N-1)\pi}{4N} & -\cos\frac{(N-1)\pi}{4N} & & \\ & & \iddots & & \ddots & & \\ & \sin\frac{3\pi}{4N} & & & & -\cos\frac{3\pi}{4N} & \\ \sin\frac{\pi}{4N} & & & & & & -\cos\frac{\pi}{4N} \end{pmatrix},$$

and P_N is a permutation matrix which when applied to a data vector corresponds to the reordering

$$\tilde{x}_n = x_{2n}, \;\; \tilde{x}_{N-1-n} = x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.88}$$

The N-point DCT-IV is decomposed into the $\frac{N}{2}$-point DCT-III and $\frac{N}{2}$-point DST-III. If it is necessary, the matrices $C^{\text{III}}_{\frac{N}{2}}$ and $S^{\text{III}}_{\frac{N}{2}}$ in (4.87) can be further decomposed using the transposed versions of sparse matrix factorizations (4.50) or (4.51) in Section 4.4.3. The generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ is shown in Fig. 4.17.

An alternative sparse matrix factorization of the DCT-IV matrix is defined as [52]

$$C^{\text{IV}}_N = P_N \begin{pmatrix} \sqrt{2} & & & \\ & I_{\frac{N}{2}-1} & & -J_{\frac{N}{2}-1} \\ & & -\sqrt{2} & \\ & -J_{\frac{N}{2}-1} & & -I_{\frac{N}{2}-1} \end{pmatrix} \begin{pmatrix} C^{\text{II}}_{\frac{N}{2}} J_{\frac{N}{2}} & 0 \\ 0 & C^{\text{II}}_{\frac{N}{2}} D_{\frac{N}{2}} \end{pmatrix} G_N \begin{pmatrix} J_{\frac{N}{2}} & 0 \\ 0 & -J_{\frac{N}{2}} \end{pmatrix}, \tag{4.89}$$

where P_N is a permutation matrix which when applied to a data vector corresponds to the reordering

$$\tilde{x}_n = x_{2n}, \;\; \tilde{x}_{N-1-n} = -x_{2n+1}, \quad n = 0, 1, \ldots, \frac{N}{2} - 1. \tag{4.90}$$

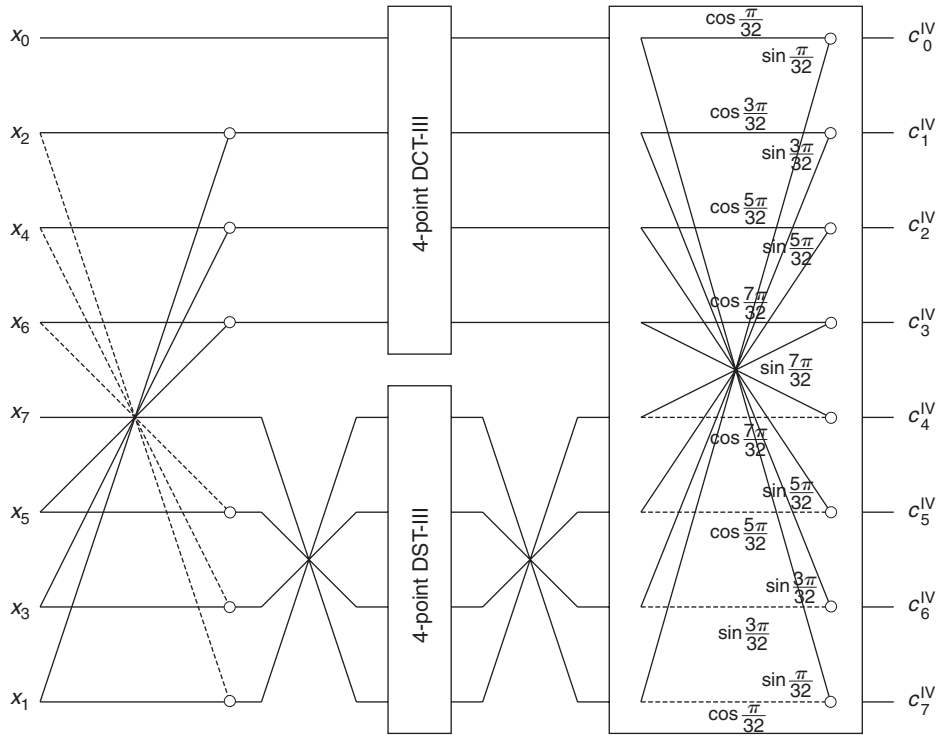


Fig. 4.17. The generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ based on (4.87).

$D_{\frac{N}{2}} = \mathrm{diag}\{(-1)^k\}, k = 0, 1, \ldots, \frac{N}{2} - 1$, is the diagonal odd-sign changing matrix, and G_N is the Givens rotation matrix

$$G_N = \begin{pmatrix} \cos\frac{(N-1)\pi}{4N} & & & & & -\sin\frac{(N-1)\pi}{4N} \\ & \cos\frac{(N-3)\pi}{4N} & & & -\sin\frac{(N-3)\pi}{4N} & \\ & & \ddots & \ddots & & \\ & & \cos\frac{\pi}{4N} & -\sin\frac{\pi}{4N} & & \\ & & \sin\frac{\pi}{4N} & \cos\frac{\pi}{4N} & & \\ & & \ddots & \ddots & & \\ & \sin\frac{(N-3)\pi}{4N} & & & \cos\frac{(N-3)\pi}{4N} & \\ \sin\frac{(N-1)\pi}{4N} & & & & & \cos\frac{(N-1)\pi}{4N} \end{pmatrix}.$$

Hence, the N-point DCT-IV is decomposed into the two $\frac{N}{2}$-point DCT-II. Again, the matrices $C_{\frac{N}{2}}^{\mathrm{II}}$ can be further decomposed using the sparse matrix factorizations (4.50) or

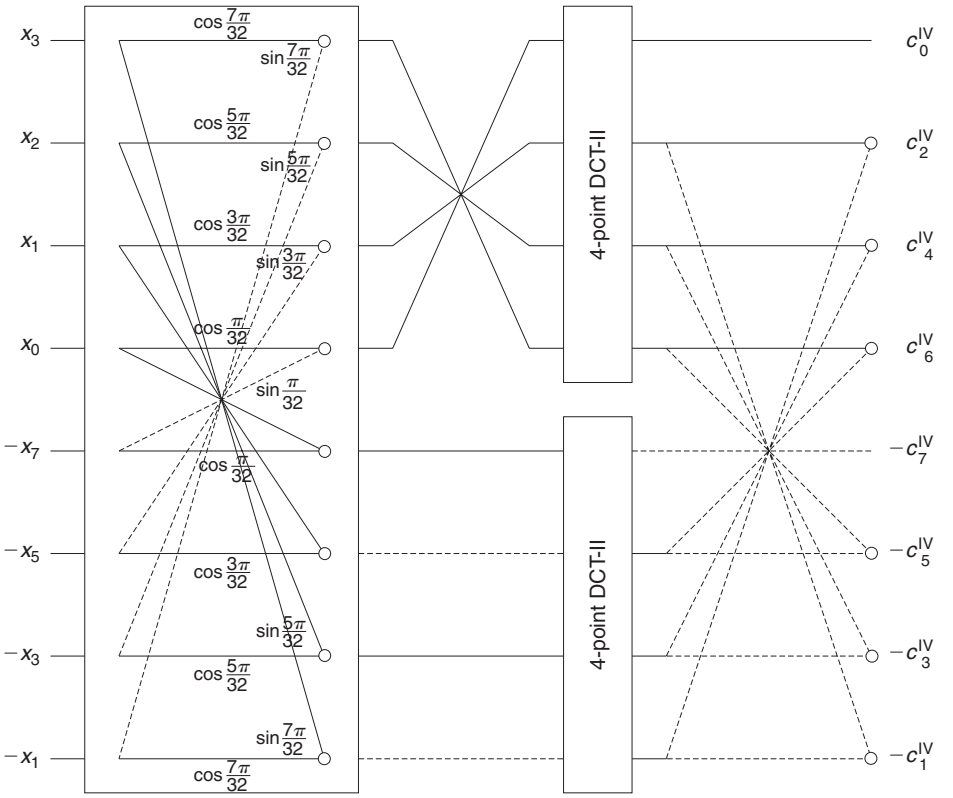


Fig. 4.18. The generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ based on (4.89).

(4.51) in Section 4.4.3. The corresponding generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ is shown in Fig. 4.18.

The orthogonal recursive sparse matrix factorization of C_N^{IV} matrix with scaling $\sqrt{2}$ has been introduced in Ref. [55]. For $N = 2^m$, $m > 1$, the DCT-IV transform matrix C_N^{IV} can be factorized into

$$C_N^{\mathrm{IV}} = P_N^{\mathrm{T}} \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & 0 \\ & I_{\frac{N}{2}-1} & I_{\frac{N}{2}-1} & \\ & I_{\frac{N}{2}-1} & -I_{\frac{N}{2}-1} & \\ 0 & & & -\sqrt{2} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & D_{\frac{N}{2}} J_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{II}} \end{pmatrix}$$

$$\times \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & D_{\frac{N}{2}} \end{pmatrix} R_N, \tag{4.91}$$

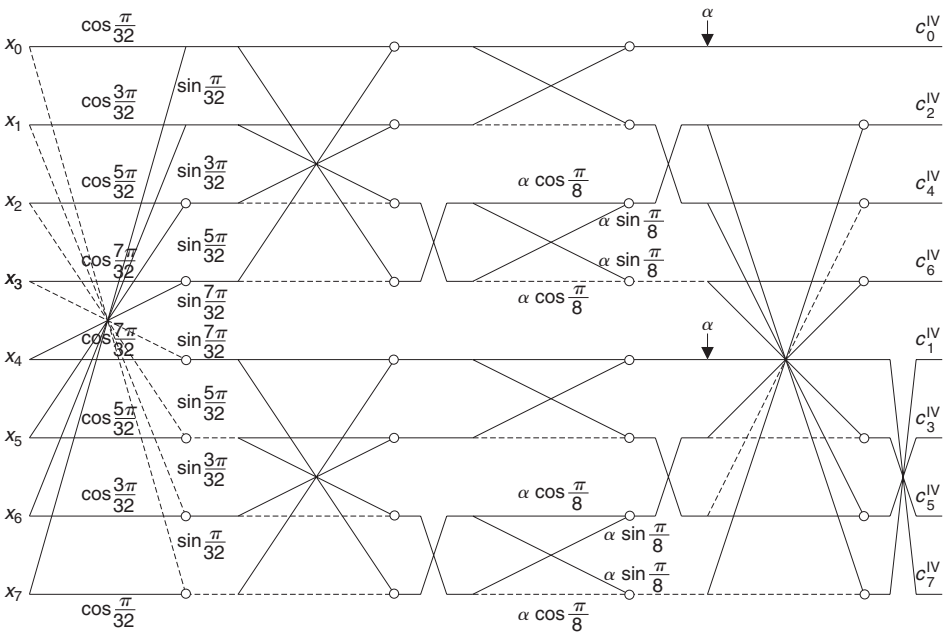


Fig. 4.19. The generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ based on (4.91); $\alpha = \sqrt{2}$.

where R_N is the Givens rotation matrix

$$R_N = \begin{pmatrix} \cos\frac{\pi}{4N} & & & & & & & \sin\frac{\pi}{4N} \\ & \cos\frac{3\pi}{4N} & & & & & \sin\frac{3\pi}{4N} & \\ & & \ddots & & & \iddots & & \\ & & & \cos\frac{(N-1)\pi}{4N} & \sin\frac{(N-1)\pi}{4N} & & & \\ & & & -\sin\frac{(N-1)\pi}{4N} & \cos\frac{(N-1)\pi}{4N} & & & \\ & & \iddots & & & \ddots & & \\ & -\sin\frac{3\pi}{4N} & & & & & \cos\frac{3\pi}{4N} & \\ -\sin\frac{\pi}{4N} & & & & & & & \cos\frac{1\pi}{4N} \end{pmatrix},$$

$D_{\frac{N}{2}} = \text{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$, is the diagonal odd-sign changing matrix, and P_N is a permutation matrix defined by (4.54). The corresponding generalized signal flow graph for the forward and inverse DCT-IV computation for $N = 8$ is shown in Fig. 4.19. For $N = 8$, each output coefficient should be normalized by a scaling factor $\frac{\sqrt{2}}{4}$ to get the true DCT-IV transform coefficients.

Note: The DST-IV sparse matrix factorizations and corresponding fast DST-IV algorithms are simply obtained from the relation between DCT-IV and DST-IV transform matrices given by (4.12).

4.5 Fast 2-D DCT/DST algorithms

In the previous Section (Section 4.4), the fast algorithms for computation of 1-D DCT and 1-D DST transforms have been presented. However, in digital image/video processing, compression and transform-based coding applications, fast 2-D algorithms (for the DCT-II, in particular) are more significant than 1-D ones. The forward and inverse 2-D DCT/DST transforms can be respectively written in matrix form as

$$Y = A_N X A_N^{\mathrm{T}}, \quad X = A_N^{\mathrm{T}} Y A_N, \tag{4.92}$$

where X is the $N \times N$ input data matrix, A_N is the DCT/DST orthogonal/orthonormal transform matrix of order N, and Y is the $N \times N$ output matrix of transform coefficients. Generally, there are two approaches to compute the 2-D DCT and 2-D DST: indirect and direct. In the indirect approach, the 2-D DCT/DST computation can be realized via other 2-D discrete orthogonal transforms such as the DFT or WHT of the same size. There are two methods of direct approach which are based on the direct 2-D DCT/DST computation. The first, the so-called row–column method, utilizes the separability property of 2-D DCT/DST transform kernels and sequentially applies any fast 1-D DCT/DST algorithm first to the rows of the input data block, and then transposes immediate results, followed by applying 1-D DCT/DST algorithm to the columns of the transformed data block. Thus, for an $N \times N$ data block, where $N = 2^n$, the row–column method requires totally $2N$ 1-D N-point transforms to be computed, and its computational complexity is $N^2 \log_2 N$ multiplications and $3N^2 \log_2 N - 2N(N-1)$ additions. The second is a 2-D vector-radix method [57] which uses 2-D decomposition process. An algorithm obtained by this method outperforms the row–column method in computational efficiency and works directly on 2-D data blocks.

4.5.1 Existing fast direct 2-D DCT-II algorithms

Since the 2-D DCT-II (typically 4×4, 8×8 and 16×16) is the standard decorrelation transform in the international image/video coding standards [6] it is not suprising that research efforts have been concentrated to develop algorithms for the efficient computation of 2-D DCT-II only. The orthonormal 2-D DCT-II for an $N \times N$ input data matrix $\{x_{mn}\}$, $m, n = 0, 1, \ldots, N-1$ is defined by the following relation [4, 6]:

$$c_{kl}^{\mathrm{II}} = \frac{2}{N} \epsilon_k \epsilon_l \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x_{mn} \cos\left[\frac{\pi(2m+1)k}{2N}\right] \cos\left[\frac{\pi(2n+1)l}{2N}\right], \quad k, l = 0, 1, \ldots, N-1, \tag{4.93}$$

and the inverse 2-D DCT-II, the 2-D DCT-III, as

$$x_{mn} = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} \epsilon_k \epsilon_l c_{kl}^{\mathrm{II}} \cos\left[\frac{\pi(2m+1)k}{2N}\right] \cos\left[\frac{\pi(2n+1)l}{2N}\right], \quad m, n = 0, 1, \ldots, N-1, \tag{4.94}$$

where

$$\epsilon_p = \begin{cases} \frac{1}{\sqrt{2}} & p = 0, \\ 1 & \text{otherwise.} \end{cases}$$

Over the last three decades many fast algorithms for the direct 2-D DCT-II computation have been developed [57–72]. Early proposed fast indirect and direct 2-D DCT-II algorithms (before 1990) are described in detail in Refs. [4, 6], and some of them (short-length DCT-II transforms) have been implemented in hardware or VLSI chips. An overview of computational complexity (multiplicative and additive) of 1-D and 2-D DCT-II algorithms together with possible improvements, scaling and implementation issues optimized for transform-based coding is discussed in Ref. [44].

Among existing fast direct 2-D radix-2 ($N = 2^n$) DCT-II algorithms, 8×8 DCT-II algorithms based on algebraic properties of the DCT-II transform matrix [62], $N \times N$ DCT-II algorithms based on polynomial transforms [64, 65] and a new fast $N \times N$ DCT-II algorithm [66–69] and its refined version in terms of regularity [70, 71] are probably the most efficient algorithms in terms of computational complexity known up to now.

The fast algorithms for the direct 8×8 DCT-II computation [62] are derived using an algebraic and computational theoretical approach. First, a matrix factorization of DCT-II transform matrix C_8^{II} is converted (with additions and permutations) to a direct sum of matrices corresponding to certain polynomial products modulo irreducible polynomials. Then, these constructions using theorems regarding the structure of Kronecker products of matrices are exploited to derive efficient 8×8 DCT-II algorithms. Although a practical fast algorithm for the 8×8 DCT-II computation requires 94 multiplications and 454 additions, its computational structure is rather complicated.

The fast direct 2-D DCT-II algorithms based on polynomial transforms [64, 65] convert the $N \times N$ DCT-II into N 1-D N-point DCTs-II and additions. In order to apply the polynomial transform technique, the $N \times N$ DCT-II is mapped by a permutation into the odd-time 2-D DFT of the same size which is inverted to an evaluation of a complex polynomial having certain symmetries. The direct polynomial transform approach reduces the number of multiplication to 50% compared to the conventional row–column method. The computation of 8×8 DCT-II by improved polynomial transform approach [65] requires 96 multiplications and 466 additions. However, the mathematical computational structure of polynomial transforms is rather complicated, especially for larger sizes. Moreover, arithmetics of complex numbers are required.

In the new direct 2-D DCT-II algorithm [66–69], the $N \times N$ DCT-II with data reordering is mapped into N 1-D N-point DCTs-II, pre- and post-addition regular butterfly structures. The main idea of the algorithm is to derive the transform kernel of the 2-D DCT-II in the form $\cos(\alpha)\,\cos(\beta) = \frac{1}{2}[\cos(\alpha - \beta) + \cos(\alpha + \beta)]$. All the multiplications are required only for the computation of 1-D N-point DCT-II. In the implementation of 1-D DCT-II any existing fast algorithm can be used. The corresponding signal flow graphs for the forward and inverse 4×4 DCT-II computation are shown in Figs. 4.20 and 4.21, respectively. The signal flow graph for the 8×8 DCT-II computation can be found in Ref. [68]. The post-addition stage of the algorithm can be further improved in terms of regularity [66, 69], at the cost of increasing the number of additions. The computational complexity of the algorithm consists of $\frac{N^2}{2}\log_2 N$ multiplications and $\frac{5N^2}{2}\log_2 N - 2N(N-1)$ additions. Thus, the 8×8 DCT-II computation requires 96 multiplications and 466 additions. Compared to polynomial transform-based algorithm the new 2-D DCT-II algorithm has the advantage in that the computational structure is highly regular and systematic, and only real arithmetic is required.

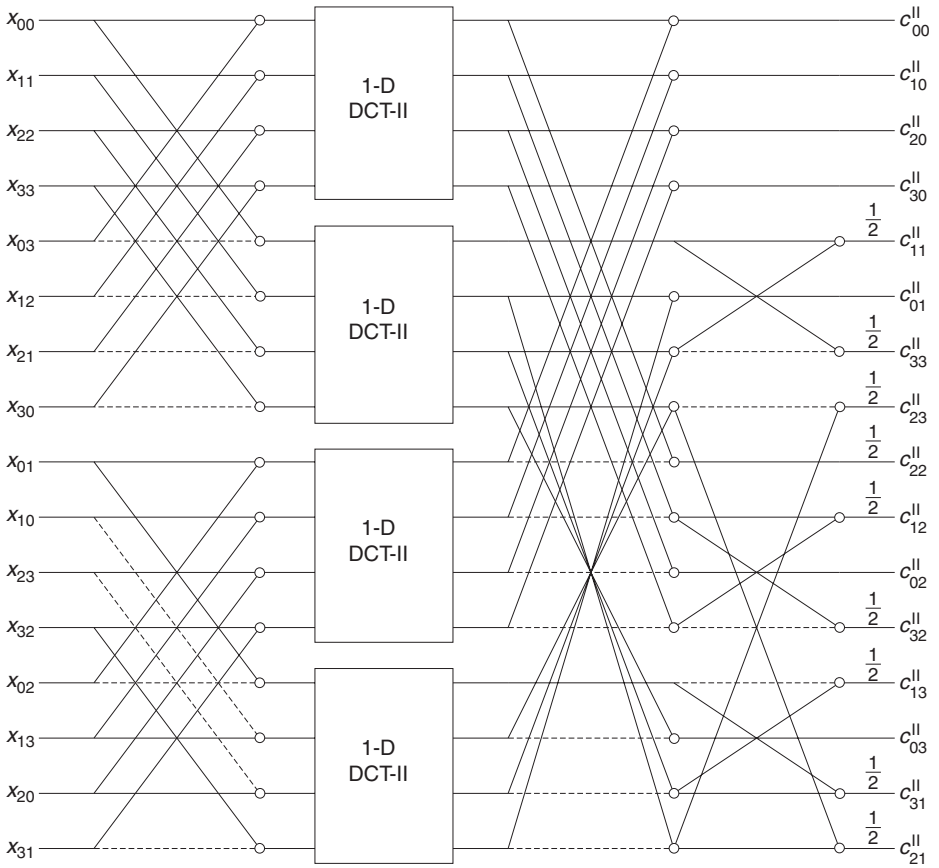


Fig. 4.20. The signal flow graph for the forward 4×4 DCT-II computation based on Refs. [66–69]. © 1991, IEEE.

A refined version of the direct 2-D DCT-II algorithm [68] has been presented in Ref. [70, 71]. Using a new proposed index-permutation the $N \times N$ DCT-II is mapped into N 1-D N-point DCTs-II with a post-addition butterfly stage which is more regular than that of the original algorithm. For illustration, the signal flow graph for the 4×4 DCT-II computation is shown in Fig. 4.22. The refined signal flow graph for the 8×8 DCT-II computation can be found in Refs. [70, 71]. It is interesting to note that if we consider two signal flow graphs for 4×4 DCT-II computations shown in Refs. [70] and [71] separately, it is observed that by exchanging indices of 2-D elements in the input and output data blocks, i.e., x_{mn} and c^{II}_{kl} by x_{nm} and c^{II}_{lk}, the algorithm remains valid. This fact indicates a certain symmetry of the algorithm.

In principle, all the above fast direct 2-D DCT-II algorithms have the same multiplicative complexity and some structural similarities. These similarities are best summarized by the fact that the $N \times N$ DCT-II has the same multiplicative complexity as N 1-D N-point

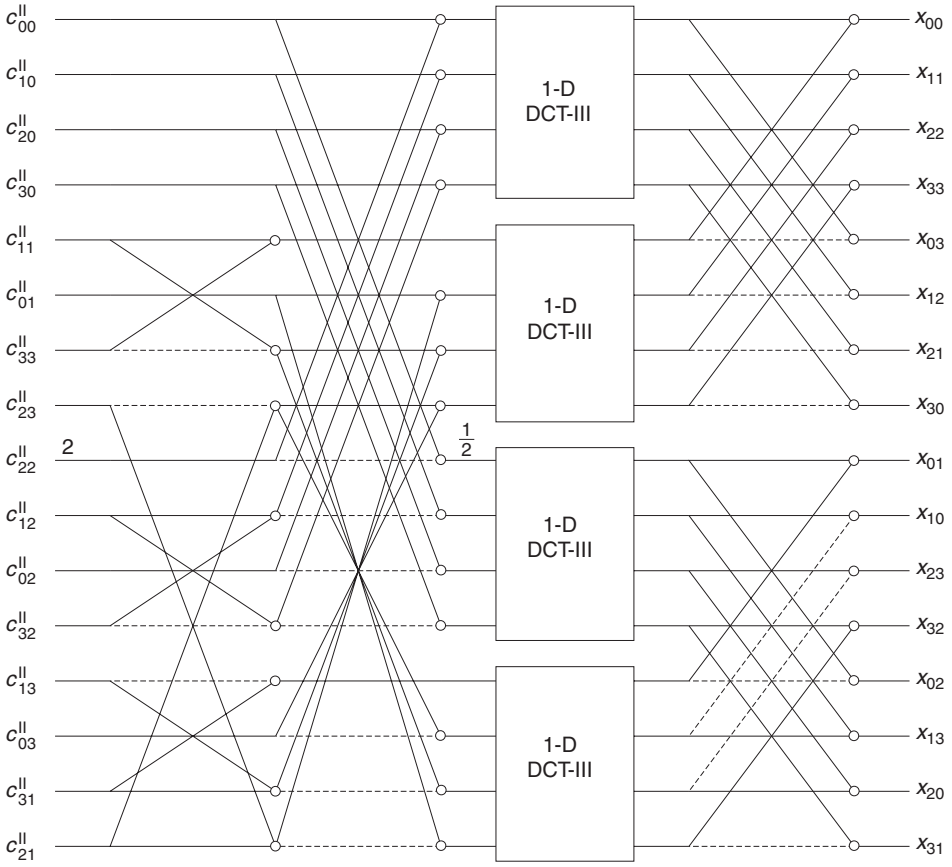


Fig. 4.21. The signal flow graph for the inverse 4×4 DCT-II computation based on Refs. [66–69]. © 1991, IEEE.

DCTs-II. On the other hand, as was noted in Ref. [65] this fact leads to an important conclusion: if an optimal algorithm is obtained for the 1-D DCT-II, then the corresponding direct 2-D DCT-II algorithms [65, 68, 71] will also be optimal. In general, the idea of the above efficient direct 2-D algorithms can be applied to any DCT/DST transform.

Another class of fast direct 2-D DCT-II algorithms with only moderate arithmetic complexity having highly regular structure and in-place implementation has been formulated by vector-radix method [57–61]. In the vector-radix method [57], $(N \times N)$-point DCT-II is decomposed into a sum of four $(\frac{N}{2} \times \frac{N}{2})$-point DCTs-II, namely, even–even, even–odd, odd–even and odd–odd indexed elements of the data block. The decomposition process is recursively repeated until trivial (2×2)-point DCT-II remains. The resulting signal flow graph has a simple and regular computational structure. The computational complexity of the $N \times N$ DCT-II algorithm derived by vector-radix method consists of $\frac{3}{4}N^2 \log_2 N$ multiplications and $3N^2 \log_2 N - 2N(N-1)$ additions, i.e., it saves 25% multiplications compared to the conventional row–column method.

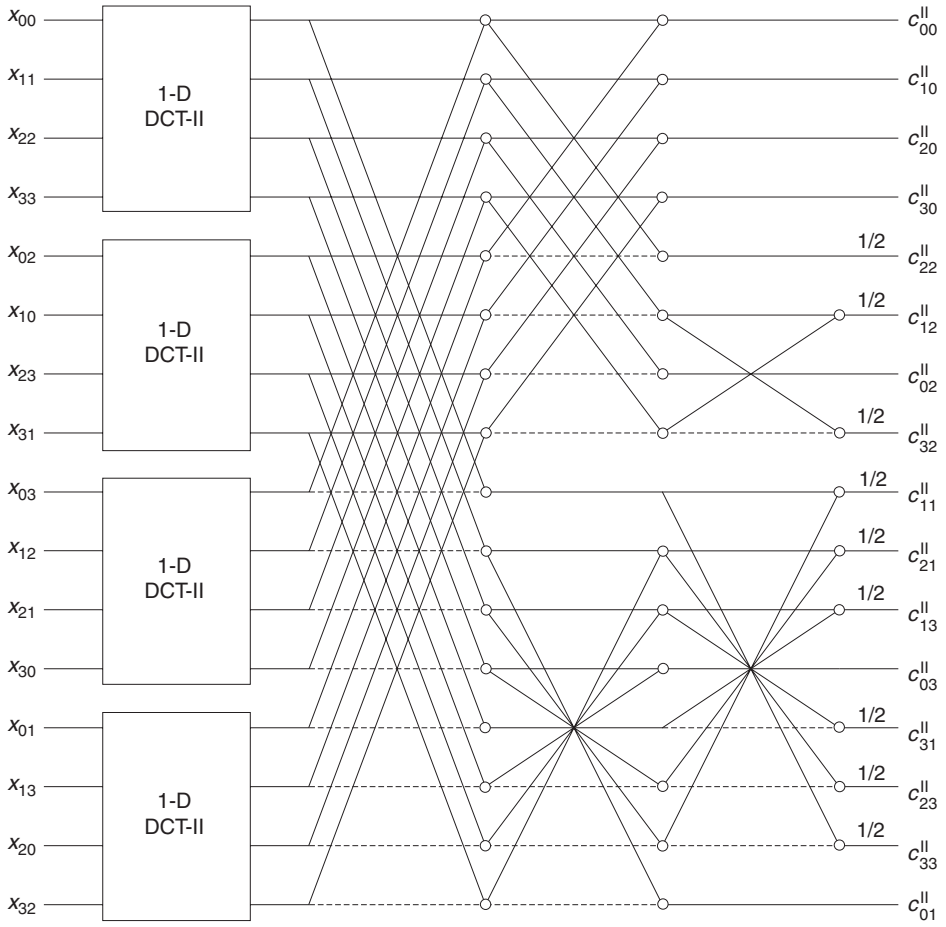


Fig. 4.22. The signal flow graph for the 4×4 DCT-II computation based on Refs. [70, 71]. © 1998, IEEE.

A structural approach is similar to the vector-radix method. The structural approach enables the derivation of a 2-D DCT-II algorithm directly from the corresponding 1-D DCT-II algorithm with the same reduction in the number of multiplications as the vector-radix method [58–61]. Mathematical derivation of the direct 2-D DCT-II algorithm is quite simple and straightforward provided by a known sparse (recursive) matrix factorization of the transform matrix. The matrix form of 1-D algorithm is extended to the 2-D matrix form using Kronecker product of matrices as a construction tool. The resulting fast direct 2-D algorithm is highly structured and simple both for hardware and software implementation. The 2-D generalized signal flow graph preserves simple, regular and systematic computational structure and reveals one-to-one correspondence between 2-D algorithm and its 1-D counterpart.

In Section 4.4, the fast 1-D DCT/DST algorithms defined by recursive sparse matrix factorizations of transform matrices have been discussed. Therefore, the structural approach

can simply be applied to any DCT and DST transform to derive its fast direct 2-D algorithm. With respect to equation (4.92), if the indices of elements of 2-D data blocks X and Y are arranged in lexicographical order of their representation, i.e., they are represented as a column vector obtained by concatenating their rows, then the $N \times N$ 2-D transform using Kronecker product of matrices can be represented in the matrix form as

$$Y = (A_N \otimes A_N)X. \tag{4.95}$$

where $\otimes$ denotes Kronecker product of matrices. The structural approach is a simple and straightforward method to generate the fast direct 2-D DCT/DST algorithm from the corresponding 1-D one.

4.5.2 The optimal 1-D 8-point and 2-D 8 × 8 DCT-II algorithms

The theoretical lower bound on the multiplicative complexity of the 1-D 2^n-point DCT-II is given by [44, 46, 56]

$$\mu(C_N^{\mathrm{II}}) = 2^{n+1} - n - 2, \tag{4.96}$$

whereas the theoretical lower bound on the multiplicative complexity of the 2-D $(2^n \times 2^n)$-point DCT-II is given by [56]

$$\mu(C_N^{\mathrm{II}} \otimes C_N^{\mathrm{II}}) = 2^n(2^{n+1} - n - 2), \tag{4.97}$$

where μ denotes the minimum number of nonrational multiplications required to perform the DCT-II transform which is represented by the $N \times N$ matrix $\hat{C}_N^{\mathrm{II}}$, and $N = 2^n$; $\otimes$ denotes the Kronecker product of matrices.

Let $\hat{C}_8^{\mathrm{II}}$ be the scaled DCT-II transform matrix with its rows rearranged in bit-reversed order. The fast algorithm for 1-D 8-point scaled DCT-II computation [45] is defined by the recursive sparse matrix factorization as

$$\hat{C}_8^{\mathrm{II}} = \frac{\sqrt{2}}{4} \begin{pmatrix} \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & 0 \\ 0 & \sqrt{2}J_2C_2^{\mathrm{IV}}J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \sqrt{2}J_4\hat{C}_4^{\mathrm{IV}}J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{4.98}$$

where

$$\sqrt{2}C_2^{\mathrm{II}} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \sqrt{2}J_2C_2^{\mathrm{IV}}J_2 = \sqrt{2}\begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix}. \tag{4.99}$$

The sparse matrix factorization of $\sqrt{2}\,J_4\hat{C}_4^{\mathrm{IV}}J_4$ is given by

$$\begin{aligned}
&\sqrt{2}J_4\hat{C}_4^{\mathrm{IV}}J_4 \\
&= \sqrt{2}\begin{pmatrix} -\cos\frac{\pi}{4} & 0 & 0 & \sin\frac{\pi}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\frac{\pi}{4} & 0 & 0 & \cos\frac{\pi}{4} \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\frac{3\pi}{16} & 0 & 0 & \sin\frac{3\pi}{16} \\ 0 & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & 0 \\ 0 & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & 0 \\ -\sin\frac{3\pi}{16} & 0 & 0 & \cos\frac{3\pi}{16} \end{pmatrix} \\
&= \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}\begin{pmatrix} \cos\frac{3\pi}{16} & 0 & 0 & \sin\frac{3\pi}{16} \\ 0 & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & 0 \\ 0 & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & 0 \\ -\sin\frac{3\pi}{16} & 0 & 0 & \cos\frac{3\pi}{16} \end{pmatrix}.
\end{aligned} \tag{4.100}$$

The signal flow graph for the fast 1-D 8-point scaled DCT-II computation is shown in Fig. 4.10.

The computation of 8-point scaled DCT-II requires 11 multiplications and 29 additions. The algorithm achieves the theoretical lower bound of the number of multiplications defined by (4.96). Therefore it is the optimal 1-D 8-point DCT-II algorithm in terms of multiplicative complexity. Using the optimal 1-D 8-point scaled DCT-II algorithm in the 2-D 8×8 DCT-II algorithm proposed in Ref. [68] and refined in Refs. [70, 71] results in an 8×8 DCT-II algorithm optimal in multiplicative complexity [72]. The resulting number of multiplications will be $8 \times 11 = 88$, which is the same as the theoretical lower bound defined by (4.97), while the number of additions remains the same. Both the algorithms have very regular computational structures and this fact leads to a very efficient and effective implementation of the 8×8 DCT-II in digital image/video applications. The normalization factors at the end of computation are reduced to shift operations.

4.5.3 Kronecker sum and product of matrices

Kronecker sum of matrices (or direct sum) and Kronecker product of matrices (or tensor product) are elegant and useful mathematical tools [3]:

- To simplify the representation of sparse matrix factorization of a transform matrix in the compact block matrix form.
- In generating higher-order matrices from lower-order ones.
- To define the direct 2-D fast algorithms from corresponding 1-D ones which can be readily generalized to higher dimensions (multidimensional transforms).

Let A and B be $m \times n$ and $p \times q$ matrices, respectively. Kronecker sum of two matrices A and B denoted by $\oplus$ is the block diagonal matrix

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix}, \tag{4.101}$$

of dimension $(m+p) \times (n+q)$.

Kronecker product of two matrices A and B denoted by $\otimes$ is defined as

$$A \otimes B = \begin{pmatrix} a_{11}B & a_{12}B & \dots & a_{1n}B \\ a_{21}B & a_{22}B & \dots & a_{2n}B \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}B & a_{m2}B & \dots & a_{mn}B \end{pmatrix}, \tag{4.102}$$

where $A \otimes C$ is an $(mp) \times (nq)$ matrix. Note that $A \otimes B \neq B \otimes A$. Kronecker product of matrices possessess a few useful properties:

1. $(A+B) \otimes C = A \otimes B + B \otimes C$,
2. $(A \otimes B) \otimes C = A \otimes (B \otimes C)$,
3. $\alpha(A \otimes B) = (\alpha A) \otimes B = A \otimes (\alpha B)$, where $\alpha \in R$,
4. $(A \otimes B)^{\mathrm{T}} = A^{\mathrm{T}} \otimes B^{\mathrm{T}}$,
5. $(A \otimes B)^{-1} = A^{-1} \otimes B^{-1}$, where A and B are square matrices,
6. $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$,
7. $A \otimes B = (A \otimes I)(I \otimes B)$,
8. $\prod_{k=1}^{r} (A_k \otimes B_k) = (\prod_{k=1}^{r} A_k) \otimes (\prod_{k=1}^{r} B_k)$, where A_k and B_k are square matrices,
9. $\det(A \otimes B) = (\det(A))^m \ (\det(B))^n$, where A is $m \times m$ and B is $n \times n$ square matrix,
10. $Trace(A \otimes B) = Trace(A)\ Trace\ (B)$,
11. If A and B are unitary, then $A \otimes B$ is also unitary.

4.5.4 Generating direct 2-D DCT/DST algorithms by structural approach

The structural approach generates a fast direct 2-D DCT/DST algorithm from the corresponding 1-D DCT/DST algorithm by a simple and straightforward procedure using the properties of Kronecker matrix products. For a given DCT/DST transform, we need only a (recursive) block matrix factorization of its transform matrix.

To illustrate the construction of 2-D DCT/DST algorithm by the structural approach, consider the computation of DCT-II for $N = 8$. The orthogonal recursive block matrix

factorization of the DCT-II transform matrix C_8^{II} given by (4.55) with scaling $\sqrt{2}$ is expressed as [55]

$$C_8^{\mathrm{II}} = \frac{\sqrt{2}}{4} P_8^{\mathrm{T}} \begin{pmatrix} P_4^{\mathrm{T}} & 0 \\ 0 & P_4^{\mathrm{T}} \end{pmatrix} \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & & & 0 \\ & \sqrt{2}C_2^{\mathrm{IV}} & & \\ & & \sqrt{2}C_2^{\mathrm{II}} & \\ 0 & & & \sqrt{2}C_2^{\mathrm{II}} \end{pmatrix} \times \begin{pmatrix} \sqrt{2}T_4 & 0 \\ 0 & \sqrt{2}R_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ I_4 & -J_4 \end{pmatrix}, \tag{4.103}$$

or in the modified form

$$C_8^{\mathrm{II}} = \frac{\sqrt{2}}{4} B_8 \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & & & 0 \\ & \sqrt{2}C_2^{\mathrm{IV}} & & \\ & & \sqrt{2}C_2^{\mathrm{II}} & \\ 0 & & & \sqrt{2}C_2^{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \sqrt{2}T_4 & 0 \\ 0 & \sqrt{2}R_4 \end{pmatrix} \times \begin{pmatrix} I_4 & 0 \\ 0 & J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{4.104}$$

where the matrix B_8 given by

$$B_8 = P_8^{\mathrm{T}} \begin{pmatrix} P_4^{\mathrm{T}} & 0 \\ 0 & P_4^{\mathrm{T}} \end{pmatrix},$$

coincides with the bit-reversal matrix and matrices A_4, $\sqrt{2}C_2^{\mathrm{II}}$, $\sqrt{2}C_2^{\mathrm{IV}}$, $\sqrt{2}T_4$ and $\sqrt{2}R_4$ are respectively given by

$$A_4 = \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & \sqrt{2} & 0 \end{pmatrix},$$

$$\sqrt{2}C_2^{\mathrm{II}} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}, \quad \sqrt{2}C_2^{\mathrm{IV}} = \begin{pmatrix} I_2 & 0 \\ 0 & D_2 \end{pmatrix} \sqrt{2} \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix},$$

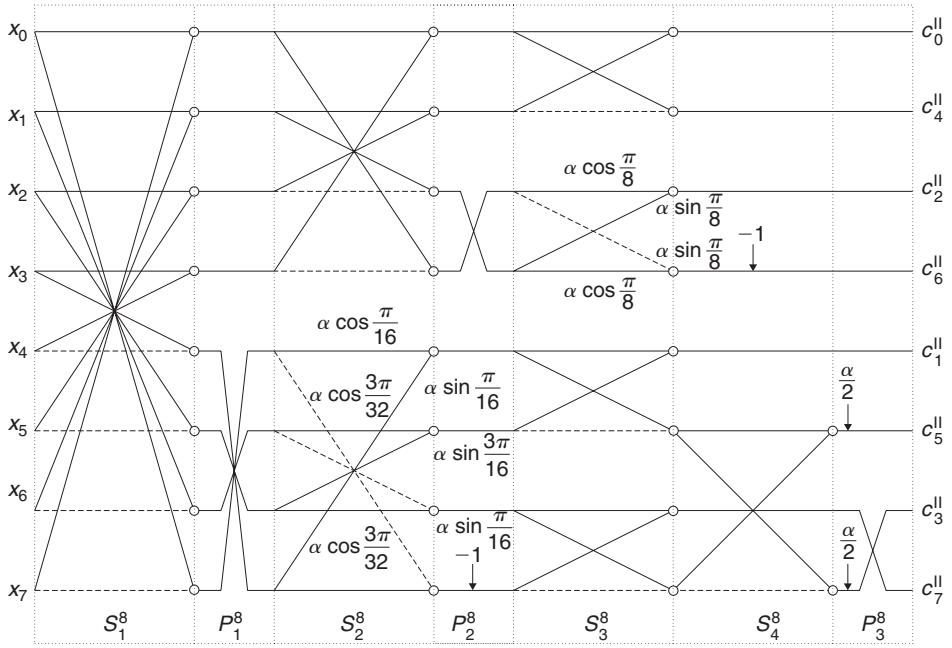


Fig. 4.23. The signal flow graph for the 8-point scaled DCT-II computation based on (4.104).

$$\sqrt{2}T_4 = \begin{pmatrix} I_2 & J_2 \\ I_2 & -J_2 \end{pmatrix} = \begin{pmatrix} I_2 & 0 \\ 0 & J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix},$$

$$\sqrt{2}R_4 = \begin{pmatrix} I_2 & 0 \\ 0 & D_2 \end{pmatrix} \sqrt{2} \begin{pmatrix} \cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \end{pmatrix}.$$

The block matrix factorization (4.103) or (4.104) defines the 1-D 8-point scaled DCT-II algorithm similar to Ref. [45]. It also requires 11 multiplications and 29 additions, achieving the theoretical lower bound for the number of multiplications and is optimal. The signal flow graph for the 1-D 8-point scaled DCT-II computation is shown in Fig. 4.23. The signal flow graph is partitioned into the blocks S_i^8, $i = 1, 2, 3, 4$, and blocks P_j^8, $j = 1, 2, 3$, which are respectively related to the butterfly and plane rotation stages of the algorithm.

Now, the fast direct 8×8 DCT-II algorithm in the matrix form is defined as

$$\begin{pmatrix} \mathbf{c}_0^{\mathrm{II}} \\ \mathbf{c}_4^{\mathrm{II}} \\ \mathbf{c}_2^{\mathrm{II}} \\ \mathbf{c}_6^{\mathrm{II}} \\ \mathbf{c}_1^{\mathrm{II}} \\ \mathbf{c}_5^{\mathrm{II}} \\ \mathbf{c}_3^{\mathrm{II}} \\ \mathbf{c}_7^{\mathrm{II}} \end{pmatrix} = (C_8^{\mathrm{II}} \otimes C_8^{\mathrm{II}}) \begin{pmatrix} \mathbf{x}_0 \\ \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \mathbf{x}_4 \\ \mathbf{x}_5 \\ \mathbf{x}_6 \\ \mathbf{x}_7 \end{pmatrix}, \tag{4.105}$$

where $\mathbf{x}_i$ and $\mathbf{c}_i^{\mathrm{II}}$ are respectively columns vectors obtained by concatenating rows of the input and output data blocks, respectively. Substituting the block matrix factorization (4.104) of the DCT-II transform matrix C_8^{II} into (4.105), and using properties of Kronecker matrix product (see properties 6 and 8 in Section 4.5.3), the fast direct 8×8 scaled DCT-II algorithm has the form

$$\begin{aligned} C_8^{\mathrm{II}} \otimes C_8^{\mathrm{II}} &= (B_8 \otimes B_8) \left\{ \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \otimes \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \right\} \\ &\times \left\{ \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & & & 0 \\ & \sqrt{2}C_2^{\mathrm{IV}} & & \\ & & \sqrt{2}C_2^{\mathrm{II}} & \\ 0 & & & \sqrt{2}C_2^{\mathrm{II}} \end{pmatrix} \otimes \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & & & 0 \\ & \sqrt{2}C_2^{\mathrm{IV}} & & \\ & & \sqrt{2}C_2^{\mathrm{II}} & \\ 0 & & & \sqrt{2}C_2^{\mathrm{II}} \end{pmatrix} \right\} \\ &\times \left\{ \begin{pmatrix} \sqrt{2}T_4 & 0 \\ 0 & \sqrt{2}R_4 \end{pmatrix} \otimes \begin{pmatrix} \sqrt{2}T_4 & 0 \\ 0 & \sqrt{2}R_4 \end{pmatrix} \right\} \left\{ \begin{pmatrix} I_4 & 0 \\ 0 & J_4 \end{pmatrix} \otimes \begin{pmatrix} I_4 & 0 \\ 0 & J_4 \end{pmatrix} \right\} \\ &\times \left\{ \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix} \otimes \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix} \right\}, \end{aligned} \tag{4.106}$$

where the direct product $B_8 \otimes B_8$ performs the 2-D permutation from bit-reverse to natural order. The detailed analysis of the structure of direct 2-D DCT-II algorithm defined by (4.104)–(4.106) results in the regular signal flow graph for the fast direct 8×8 scaled DCT-II computation shown in Fig. 4.24. In order to show one-to-one correspondence between the 2-D 8×8 DCT-II algorithm and its 1-D counterpart, the signal flow graph in Fig. 4.24 is partitioned into 2-D blocks $S_i^{8\times 8}$, $i = 1, 2, 3, 4$, and blocks $P_j^{8\times 8}$, $j = 1, 2, 3$. All blocks inscribed by S_i^8 and P_j^8 are defined in Fig. 4.23. Heavy lines represent vector operations on rows of the input data block.

If the direct 8×8 scaled DCT-II computation is realized by the row–column method, it requires 16 1-D 8-point scaled DCT-II and its computational complexity is 176

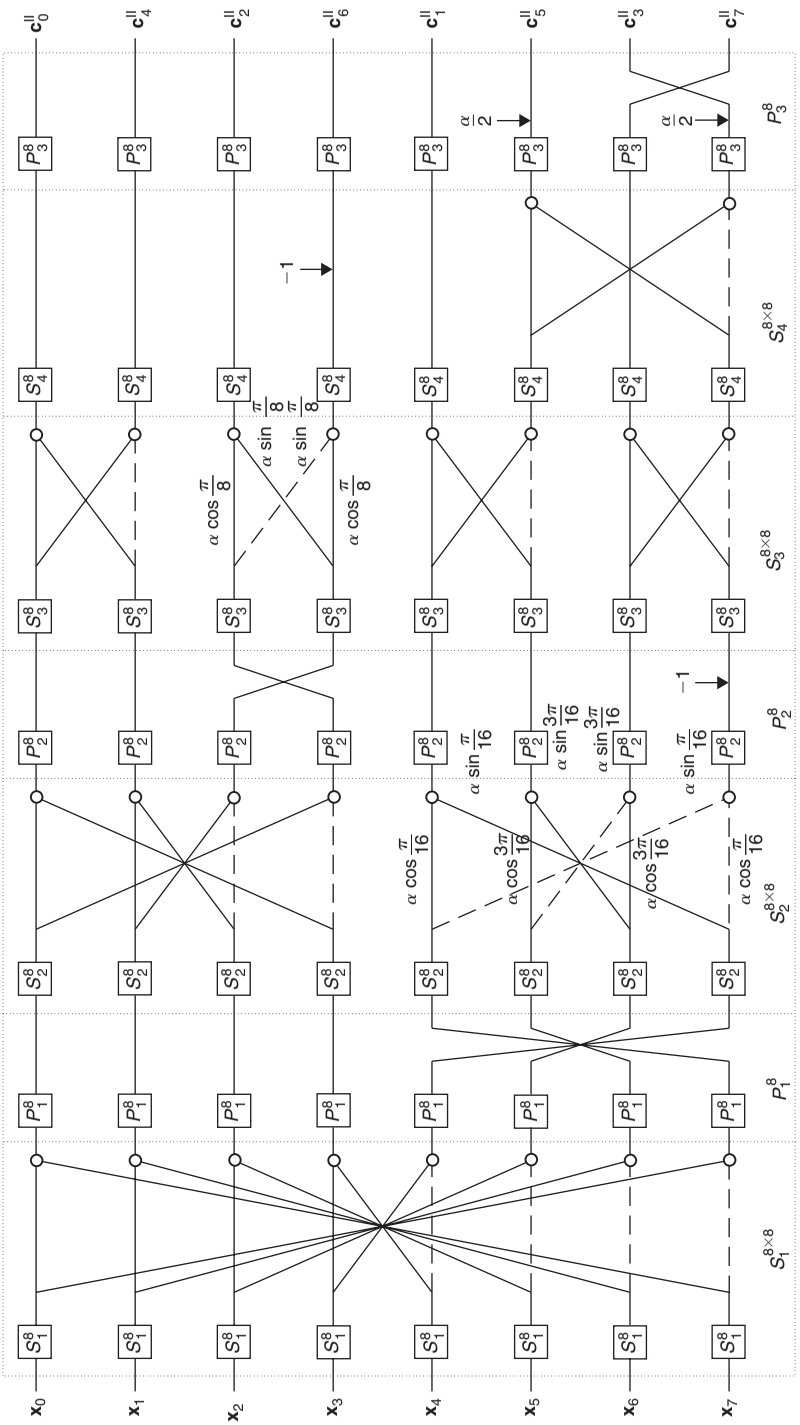


Fig. 4.24. The signal flow graph for the direct 8×8 scaled DCT-II computation based on (4.104)–(4.106).

multiplications and 464 additions plus performing the transposition of semi-transformed data block. From Fig. 4.24 it can be seen that all multiplications in the signal flow graph are contained in the blocks $S_2^{8\times 8}$, $S_3^{8\times 8}$ and S_2^8, S_3^8, including 2-D and 1-D plane rotations and in the blocks $P_3^{8\times 8}$, P_3^8 with twiddle factors $\frac{\sqrt{2}}{2}$. With respect to (4.106) 2-D plane rotations in blocks $S_2^{8\times 8}$ and $S_3^{8\times 8}$ are defined by the direct products $(\sqrt{2}R_4 \otimes \sqrt{2}R_4) = 2(R_4 \otimes R_4)$ and $(\sqrt{2}C_2^{\mathrm{IV}} \otimes \sqrt{2}C_2^{\mathrm{IV}}) = 2(C_2^{\mathrm{IV}} \otimes C_2^{\mathrm{IV}})$, respectively. The multiplications by $\frac{\sqrt{2}}{2}$ in the block $P_3^{8\times 8}$ are defined by direct product $A_4 \otimes A_4$ and they can be combined reducing four multiplications to shift operations. The total number of 1-D plane rotations to be performed by the row–column method is 48 which require 144 multiplications and the same number of additions. Each plane rotation is a Givens–Jacobi rotation defined by

$$G_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix}, \quad G_\varphi^{\mathrm{T}} = G_\varphi^{-1} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} = G_{-\varphi}, \tag{4.107}$$

where $\varphi = \frac{\pi}{8}, \frac{\pi}{16}$ and $\frac{3\pi}{16}$. In order to reduce the number of multiplications, the direct products $2(C_2^{\mathrm{IV}} \otimes C_2^{\mathrm{IV}})$ and $2(R_4 \otimes R_4)$ can be efficiently evaluated using the following useful formula [46]:

$$G_\alpha \otimes G_\beta = K_4^{\mathrm{T}} \begin{pmatrix} G_{\alpha+\beta} & 0 \\ 0 & G_{\alpha-\beta} \end{pmatrix} K_4, \quad K_4 = \begin{pmatrix} 1 & -1 & -1 & -1 \\ 1 & 1 & 1 & -1 \\ 1 & 1 & -1 & 1 \\ 1 & -1 & 1 & 1 \end{pmatrix}. \tag{4.108}$$

From the property of Kronecker matrix product (see property 4 in Section 4.5.3) we obtain the formula for the transposed direct product of Givens–Jacobi rotations $G_\alpha^{\mathrm{T}} \otimes G_\beta^{\mathrm{T}}$ as

$$(G_\alpha \otimes G_\beta)^{\mathrm{T}} = G^{\mathrm{T}}{}_\alpha \otimes G_\beta^{\mathrm{T}} = K_4^{\mathrm{T}} \begin{pmatrix} G_{\alpha+\beta}^{\mathrm{T}} & 0 \\ 0 & G_{\alpha-\beta}^{\mathrm{T}} \end{pmatrix} K_4. \tag{4.109}$$

The evaluation of direct products $(G_\alpha \otimes G_\beta)\mathbf{x}^{\mathrm{T}}$ and $(G_\alpha^{\mathrm{T}} \otimes G_\beta^{\mathrm{T}})\mathbf{x}^{\mathrm{T}}$ includes 2 plane rotations and additions only. If $\alpha \neq \beta$ it requires 6 multiplications and 22 additions. In case $\alpha = \beta$ the number of arithmetic operations is even less; if $\alpha = \beta \neq \frac{\pi}{8}$, 3 multiplications and 19 additions are required, and if $\alpha = \beta = \frac{\pi}{8}$, 2 multiplications and 18 additions are required. Using the formula (4.108), the direct product $2(C_2^{\mathrm{IV}} \otimes C_2^{\mathrm{IV}})\mathbf{x}^{\mathrm{T}}$ is reduced to the evaluation of

$$2(G_{\frac{\pi}{8}} \otimes G_{\frac{\pi}{8}})\mathbf{x}^{\mathrm{T}} \quad \text{requiring 2 multiplications and 18 additions,}$$

and direct product $2(R_4 \otimes R_4)\mathbf{x}^{\mathrm{T}}$ is reduced to the evaluation of

$$\begin{aligned}
&2(G_{\frac{\pi}{16}} \otimes G_{\frac{\pi}{16}})\mathbf{x}^{\mathrm{T}} \quad \text{requiring 3 multiplications and 19 additions,} \\
&2(G_{\frac{\pi}{16}} \otimes G_{\frac{3\pi}{16}})\mathbf{x}^{\mathrm{T}} \quad \text{requiring 6 multiplications and 22 additions,} \\
&2(G_{\frac{3\pi}{16}} \otimes G_{\frac{\pi}{16}})\mathbf{x}^{\mathrm{T}} \quad \text{requiring 6 multiplications and 22 additions,} \\
&2(G_{\frac{3\pi}{16}} \otimes G_{\frac{3\pi}{16}})\mathbf{x}^{\mathrm{T}} \quad \text{requiring 3 multiplications and 19 additions.}
\end{aligned}$$

The evaluation of direct products $2(C_2^{\mathrm{IV}} \otimes C_2^{\mathrm{IV}})$ and $2(R_4 \otimes R_4)$ by (4.108) reduces the number of 1-D Givens–Jacobi rotations to be evaluated from 48 to 35. Thus, the total arithmetic complexity for the direct 8×8 scaled DCT-II algorithm derived by the structural approach consists of 128 multiplications and 476 additions. Compared to the 8×8 DCT-II algorithm derived by the vector-radix method the multiplicative complexity of the direct 8×8 scaled DCT-II algorithm derived by the structural approach is better. The reason in the significant reduction of multiplicative complexity, but at the cost of the increased number of additions can be explained

- by using the optimal 1-D 8-point scaled DCT-II algorithm,
- by efficient evaluation of direct products $2(G_\alpha \otimes G_\beta)$ by (4.108).

The normalization of DCT-II transformed coefficients at the end of signal flow graph is reduced to shift operations.

The idea of this fast direct 8×8 DCT-II algorithm can be easily extended to any DCT and DST. Since almost all DCT/DST algorithms presented in Section 4.4 are based on the (recursive) block matrix factorization of the transform matrix, the 2-D algorithm can be directly constructed from corresponding 1-D algorithm by the structural approach.

4.6 Summary

The fast 1-D and 2-D DCT/DST algorithms for all even types of the DCT and DST have been discussed in detail. Almost all are direct algorithms defined by the (recursive, if it exists) sparse matrix factorization of the transform matrix. The definitions, basic mathematical properties and relations between corresponding DCT and DST have been briefly discussed (Section 4.2). The explicit forms of orthonormal DCT and DST matrices for $N = 2, 4$ and 8 have been presented (Section 4.3). The fast 1-D DCT/DST algorithms are numerically stable, have the regular computational structures and employ real arithmetics only (Section 4.4). In particular, these rotation-based fast algorithms are very convenient for the construction of corresponding integer transforms. The generalized signal flow graphs corresponding to the sparse matrix factorization of the transform matrix for the fast DCT/DST computation have also been provided and they are ready to be used in practical applications. Finally, in Section 4.5 it is shown how the 2-D DCT/DST fast algorithms can be derived from the corresponding 1-D ones.

Problems and Exercises

1. Using the definitions of DCT-II and DST-II prove the relation between the DCT-II and DST-II matrices given by (4.11)(see [39]).
2. Using the definitions of DCT-IV and DST-IV prove the relation between the DCT-IV and DST-IV matrices given by (4.12).
3. Using the definitions of DCT-IV and DCT-II prove the relation between the DCT-IV and DCT-II matrices given by (4.13). (Hint: decompose the DCT-IV transform

kernel using the trigonometric identity $\cos(\alpha+\beta) = 2\cos(\alpha)\cos(\beta) - \cos(\alpha-\beta)$ with the proper setting α and β.)

4. Prove the relations between DCT and DST matrices given by (4.14).
5. Based on the definitions of orthonormal DCT and DST matrices derive their explicit forms for $N = 16$.
6. Prove the even symmetry/antisymmetry property of basis vectors defined by (4.15) and (4.16) for $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$ and C_N^{II} matrices.
7. Prove the odd symmetry/antisymmetry property of basis vectors defined by (4.19) and (4.20) for C_{N+1}^{I} and S_{N-1}^{I} matrices.
8. For the matrices $\tilde{C}_N^{\mathrm{I}}$, $\tilde{S}_N^{\mathrm{I}}$ and C_N^{II}, $N = 4$ and 8, derive their EOT matrix factorizations defined by (4.17) or (4.18). If it is possible, apply EOT factorization recursively.
9. Consider the fast DCT-I computation based on the SCT algorithm defined by (4.21) and (4.22). The corresponding generalized signal flow graph for $N = 3$, 5 and 9 is shown in Fig. 4.1. Try to derive the recursive sparse matrix factorization of the DCT-I matrix C_{N+1}^{I} and verify its correctness by computer program. Recall that the sparse matrix factorization of the transform matrix defines the fast algorithm repesented by the signal flow graph and vice versa.
10. Consider the fast DCT-I computation defined by (4.25) with the corresponding generalized signal flow graph for $N = 2$, 4 and 8 shown in Fig. 4.2. Write down the sparse matrix factors and verify their correctness by computer program.
11. Develop the generalized signal flow graphs for the fast DCT-I computation for $N = 2$, 4 and 8 based on the sparse matrix factorization (4.27) and verify their correctness by computer program.
12. The generalized signal flow graph for the fast DCT-I computation based on split-radix algorithm is shown in Fig. 4.3. Investigate and verify the validity of split-radix fast DCT-I algorithm for $N = 16$ defined by the sparse matrix factorization of DCT-I matrix C_{N+1}^{I} (4.31)–(4.33) and draw the corresponding signal flow graph. Correct the proposed sparse matrix factorization if necessary.
13. Develop the generalized signal flow graphs for the fast DCT-I computation for $N = 2$, 4 and 8 based on the sparse matrix factorization (4.34) and verify their correctness by computer program.
14. For each fast DCT-I algorithm in Section 4.4.1 for $N = 2$, 4 and 8 verify its correctness by computer program and list its computational complexity (the number of multiplications and additions).
15. Extend each fast DCT-I algorithm in Section 4.4.1 to $N = 16$, i.e., derive the sparse matrix factorization of DCT-I matrix C_{16}^{I}, draw the corresponding signal flow graph and verify its correctness by computer program. List the computational complexity of algorithms. Finally, compare the algorithms in terms of structural simplicity and computational complexity.
16. Consider the fast DST-I computation based on the SST algorithm defined by (4.35) and (4.36). The corresponding generalized signal flow graph for $N = 3$ and 7 is shown

in Fig. 4.4. Try to derive the recursive sparse matrix factorization of the DST-I matrix S^{I}_{N-1} and verify its correctness by computer program.

17. Consider the fast DST-I computation defined by (4.39) with the corresponding generalized signal flow graph for $N = 4$ and 8 shown in Fig. 4.5. Write down the sparse matrix factors and verify their correctness by computer program.

18. Consider the fast DST-I computation defined by (4.40) with the corresponding generalized signal flow graph for $N = 4$ and 8 shown in Fig. 4.6. Write down the sparse matrix factors and verify their correctness by computer program.

19. Develop the generalized signal flow graphs for the fast DST-I computation for $N = 4$ and 8 based on the sparse matrix factorizations (4.42) and (4.42), and verify their correctness by computer program.

20. The generalized signal flow graph for the fast DST-I computation based on split-radix algorithm is shown in Fig. 4.7. Investigate and verify the validity of split-radix fast DST-I algorithm for $N = 16$ defined by the sparse matrix factorization of DST-I transform matrix S^{I}_{N-1} (4.47)–(4.49) and draw the corresponding signal flow graph. Correct the proposed sparse matrix factorization if necessary.

21. For each fast DST-I algorithm in Section 4.4.2 for $N = 4$ and 8 verify its correctness by computer program and list its computational complexity.

22. Extend each fast DST-I algorithm in Section 4.4.2 to $N = 16$, i.e., derive the sparse matrix factorization of DST-I matrix S^{I}_{15}, draw the corresponding signal flow graph and verify its correctness by computer program. Compare the algorithms in terms of structural simplicity and computational complexity.

23. Extend the fast DCT-II algorithm defined by (4.50) to $N = 16$. The generalized signal flow graph for $N = 2$, 4 and 8 is shown in Fig. 4.8. Write down the sparse matrix factorization and draw the corresponding signal flow graph. Verify its correctness by computer program and list the computational complexity. Because of recursivity of DCT-II matrix the upper half of the signal flow graph for $N = 16$ after the first butterfly stage will correspond to the signal flow graph for $N = 8$. Therefore it is sufficient only to derive the sparse matrix factorization of DCT-IV matrix C^{IV}_8 (see Ref. [33]). Note that the matrix C^{IV}_8 is symmetric.

24. Consider the fast DCT-II algorithm defined by (4.51) with corresponding generalized signal flow graph for $N = 2$, 4 and 8 shown in Fig. 4.9. Write down sparse matrix factors for $N = 2$, 4 and 8, verify the correctness of algorithm and list its computational complexity. Then extend the algorithm to $N = 16$ and repeat the above steps.

25. Consider the fast DCT-II algorithm defined by (4.53) and (4.55). Its generalized signal flow graph $N = 2, 4$ and 8 is shown in Fig. 4.10. Write down sparse matrix factors for $N = 2$, 4 and 8, verify the correctness of algorithm and list its computational complexity. Then extend the algorithm to $N = 16$ and repeat the above steps.

26. The fast algorithm for 16-point scaled DCT-II computation represented by the signal flow graph in Ref. [45] is not correct. Verify the signal flow graph and correct it. Verify by computer program.

27. Extend the fast DCT-II computation via WHT to $N = 16$. Develop the sparse matrix factorization and draw corresponding signal flow graph. Verify the correctness of algorithm by computer program and list its computational complexity. It is necessary to find a sparse matrix factorization of WHT matrix [1] for $N = 16$. Use the recursivity property of the conversion matrix to find a sparse matrix factorization (analytical form) of the block matrix U_8 (see Ref. [37]).

28. Verify the fast FFCT algorithm for the DCT-II computation via DFT for $N = 8$ by computer program. Its generalized signal flow graph is shown in Fig. 4.13. Derive the corresponding sparse matrix factorization of the DCT-II transform matrix C_8^{II}. Extend the FFCT algorithm to $N = 16$, write down the sparse matrix factorization, draw signal flow graph and list the computational complexity.

29. The generalized signal flow graph for the fast DCT-II computation based on split-radix algorithm is shown in Fig. 4.14. Investigate and verify the validity of split-radix fast DCT-II algorithm for $N = 16$ defined by the sparse matrix factorization of DCT-II matrix C_N^{II} (4.81) and (4.82) and draw the corresponding signal flow graph. Correct the proposed sparse matrix factorization if necessary.

30. Extend the fast DCT-IV algorithm defined by (4.83) to $N = 16$ with the improved factorization of DCT-IV matrix (equation (4.86)). The generalized signal flow graph for $N = 8$ is shown in Fig. 4.16. Write down the sparse matrix factorization and draw the corresponding signal flow graph. Verify its correctness by computer program and list the computational complexity.

31. Consider the fast DCT-IV algorithm defined by (4.87) with corresponding generalized signal flow graph for $N = 8$ shown in Fig. 4.17. Write down sparse matrix factors and verify the correctness of algorithm by computer program and list the computational complexity. Then extend the algorithm to $N = 16$ and repeat the above steps.

32. Consider the fast DCT-IV algorithm defined by (4.89) with corresponding generalized signal flow graph for $N = 8$ shown in Fig. 4.18. Write down sparse matrix factors and verify the correctness of algorithm by computer program and list the computational complexity. Then extend the algorithm to $N = 16$ and repeat the above steps.

33. Consider the fast DCT-IV algorithm defined by (4.91) with corresponding generalized signal flow graph for $N = 8$ shown in Fig. 4.19. Write down the sparse matrix factors, verify the correctness of algorithm by computer program and list the computational complexity. Then extend the algorithm to $N = 16$ and repeat the above steps.

34. Try to propose the fast split-radix DCT-IV algorithm. Derive the complete formulae, the sparse matrix factorization of the DCT-IV matrix C_N^{IV} and draw the corresponding signal flow graph for $N = 2, 4, 8$ and 16.

35. Verify by computer program the direct fast algorithm for the forward and inverse 4×4 DCT-II computation respectively represented by the signal flow graphs shown in Figs. 4.20 and 4.21. Substitute the suitable 1-D fast DCT-II algorithm.

36. Verify by computer program the refined direct fast algorithm for the forward 4×4 DCT-II computation represented by the signal flow graph shown in Fig. 4.22. Substitute the suitable 1-D fast DCT-II algorithm.

37. Implement the optimal 8-point scaled DCT-II and optimal 2-D 8×8 DCT-II algorithm in terms of multiplicative complexity for the efficient computation of DCT-II in the international coding standards [6].

38. Verify by computer program the fast algorithm for the 8-point DCT-II computation represented by signal flow graph shown in Fig. 4.23.

39. Verify the direct 2-D fast algorithm for the 8×8 DCT-II computation represented by the signal flow graph shown in Fig. 4.24. Analyze its computational complexity.

40. Prove (4.108) and (4.109).

41. Having the (recursive) sparse matrix factorizations of DCT and DST matrices for $N = 2, 4$ and 8 and by defining the 1-D fast DCT and DST algorithms try to derive subsequently the direct 2-D DCT and DST fast algorithms from corresponding 1-D ones using the structural approach. Write down the sparse matrix factorizations and draw the signal flow graphs. Verify the correctness by computer program. Compare the obtained computational complexity with the conventional row–column method. Extend to $N = 16$.

References

[1] N. Ahmed and K. R. Rao, *Orthogonal Transforms for Digital Signal Processing*, Springer-Verlag, Berlin, 1975.

[2] R. J. Clarke, *Transform Coding of Images*, Academic Press, London, 1985.

[3] A. K. Jain, *Fundamentals of Digital Image Processing*, Prentice-Hall, Englewood Cliffs, New Jersey, 1989.

[4] K. R. Rao and P. Yip, *Discrete Cosine Transform: Algorithms, Advantages, Applications*, Academic Press, Boston, 1990.

[5] H. S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Norwood, MA, 1992, Chapters 1 and 2, pp. 1–80.

[6] K. R. Rao and J. J. Hwang, *Techniques and Standards for Digital Image/Video/Audio Coding*, Prentice-Hall, Upper Saddle River, New Jersey, 1996.

[7] K. R. Rao and P. Yip, Editors, *The Transform and Data Compression Handbook*, CRC Press, Boca Raton, FL, 2001, Chapter 4: Discrete cosine and sine transforms, pp. 117–195.

[8] N. Ahmed, T. Natarajan and K. R. Rao, "Discrete cosine transform", *IEEE Transactions on Computers*, Vol. C-23, No. 1, January 1974, pp. 88–93.

[9] Z. Wang and B. R. Hunt, "The discrete cosine transform – a new version", *Proceedings of the IEEE ICASSP'83*, Boston, MA, April 1983.

[10] H. Kitajima, "A symmetric cosine transform", *IEEE Transactions on Computers*, Vol. C-29, April 1980, pp. 317–323.

[11] A. K. Jain, "A fast Karhunen–Loève transform for a class of random processes", *IEEE Transactions on Communications*, Vol. COM-24, September 1976, pp. 1023–1029.

[12] H. B. Kekre and J. K. Solanki, "Comparative performance of various trigonometric unitary transforms for transform image coding", *International Journal of Electronics*, Vol. 44, No. 3, 1978, pp. 305–315.

[13] A. K. Jain, "A sinusoidal family of unitary transforms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-1, October 1979, pp. 356–365.

[14] Z. Wang and B. R. Hunt, "The discrete W transform", *Applied Mathematics and Computation*, Vol. 16, January 1985, pp. 19–48.

[15] G. Strang, "The discrete cosine transform", *SIAM Review*, Vol. 41, No. 1, March 1999, pp. 135–147.

[16] V. Sanchez, P. Garcia, A. M. Peinado, J. C. Segura and A. J. Rubio, "Diagonalizing properties of the discrete cosine transforms", *IEEE Transactions on Signal Processing*, Vol. 43, No. 11, November 1995, pp. 2631–2641.

[17] V. Sanchez, A. M. Peinado, J. C. Segura, P. Garcia and A. J. Rubio, "Generating matrices for the discrete sine transforms", *IEEE Transactions on Signal Processing*, Vol. 44, No. 10, October 1996, pp. 2644–2646.

[18] S. A. Martucci, "Symmetric convolution and the discrete sine and cosine transforms", *IEEE Transactions on Signal Processing*, Vol. 42, No. 5, May 1994, pp. 1038–1051.

[19] G. Bongiovanni, P. Corsini and G. Frosini, "One-dimensional and two-dimensional generalized Fourier transforms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-24, No. 1, February 1976, pp. 97–99.

[20] N. C. Hu, H. I. Chang and O. K. Ersoy, "Generalized discrete Hartley transforms", *IEEE Transactions on Signal Processing*, Vol. 40, No. 12, December 1992, pp. 2931–2940.

[21] Z. Wang, Comments on generalized discrete Hartley transform", *IEEE Transactions on Signal Processing*, Vol. 43, No. 7, July 1995, pp. 1711–1712.

[22] Z. Wang, G. A. Julien and W. C. Miller, "The generalized discrete W transform and its application to interpolation", *Signal Processing*, Vol. 36, 1994, pp. 99–109.

[23] V. Britanak and K. R. Rao, "The fast generalized discrete Fourier transform: a unified approach to the discrete sinusoidal transforms computation", *Signal Processing*, Vol. 79, No. 12, December 1999, pp. 135–150.

[24] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm", *Electronics Letters*, Vol. 20, No. 1, January 1984, pp. 14–16.

[25] H. V. Sorensen, M. T. Heideman and C. S. Burrus, "On computing the split-radix FFT", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 1, Febuary 1986, pp. 152–156.

[26] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 2, April 1986, pp. 285–295.

[27] C. W. Sun and P. Yip, "Split-radix algorithms for DCT and DST", *Proceedings of the Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 1989, pp. 508–512.

[28] P. Yip and K. R. Rao, "On the computation and effectiveness of discrete sine transform", *Computers and Electrical Engineering*, Vol. 7, No. 1, 1980, pp. 45–55.

[29] P. Yip and K. R. Rao, "A fast computational algorithm for the discrete sine transform", *IEEE Transactions on Communications*, Vol. COM-28, No. 2, February 1980, pp. 304-307.

[30] E. J. K. Bisherurwa and F. P. Coakley, "New fast discrete sine transform with offset", *Electronics Letters*, Vol. 17, No. 21, October 1981, pp. 803–805.

[31] Z. Wang, "Comments on A fast computational algorithm for the discrete sine transform", *IEEE Transactions on Communications*, Vol. COM-34, No. 2, February 1986, pp. 204–205.

[32] V. Britanak, "A unified approach to the fast computation of discrete sinusoidal transforms I: DCT and DST transforms", *Computers and Artificial Intelligence*, Vol. 17, No. 6, December 1998, pp. 583–607.

[33] W. H. Chen, C. H. Smith and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform", *IEEE Transactions on Communications*, Vol. COM-25, September 1977, pp. 1004–1009.

[34] D. Hein and N. Ahmed, "On a real-time Walsh–Hadamard/cosine transform image processor", *IEEE Transactions on Electromagnetic Compatibility*, Vol. EMC-20, No. 3, August 1978, pp. 453–457.

[35] Y. J. Chen, S. Oraintara and T. Nguyen, "Integer discrete cosine transform (IntDCT)", *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing*, Singapore, December 1999.

[36] Z. Wang, "Reconsideration of a fast computational algorithm for the discrete cosine transform", *IEEE Transactions on Communications*, Vol. COM-31, January 1983, pp. 121–123.

[37] S. Venkataraman, V. R. Kanchan, K. R. Rao and M. Mohanty, "Discrete transforms via the Walsh–Hadamard transform", *Signal Processing*, Vol. 14, No. 4, June 1988, pp. 371–382.

[38] R. Srinivasan and K. R. Rao, "An approximation to the discrete cosine transform for $N = 16$", *Signal Processing*, Vol. 5, January 1983, pp. 81–85.

[39] Z. Wang, "A fast algorithm for the discrete sine transform implemented by the fast cosine transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-30, October 1982, pp. 814–815.

[40] Z. Wang, "Fast algorithms for the discrete W transform and for the discrete Fourier transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-32, August 1984, pp. 803–816.

[41] N. Suehiro and M. Hatori, "Fast algorithms for the discrete Fourier transform and other transforms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34, June 1986, pp. 642–644.

[42] M. Vetterli and H. J. Nussbaumer, "Simple FFT and DCT algorithms with reduced number of operations", *Signal Processing*, Vol. 6, August 1984, pp. 267–278.

[43] M. Vetterli and A. Ligtenberg, "A discrete Fourier-cosine transform chip", *IEEE Journal of Selected Areas in Communications*, Vol. SAC-4, January 1986, pp. 49–61.

[44] P. Duhamel and H. H'mida, "New 2^n DCT algorithms suitable for VLSI implementation", *Proceedings of the IEEE ICASSP'87*, Dallas, TX, April 1987, pp. 1805–1808.

[45] C. Loeffler, A. Ligtenberg and G. S. Moshytz, "Practical fast 1-D DCT algorithms with 11 multiplications", *Proceedings of the IEEE ICASSP'89*, Glasgow, Scotland, May 1989, pp. 988–991.

[46] M. Vetterli, P. Duhamel and C. Guillemot, "Trade-off's in the computation of mono- and multi-dimensional DCTs", *Proceedings of the IEEE ICASSP'89*, Glasgow, Scotland, May 1989, pp. 999–1002.

[47] Z. Wang, "On computing the discrete Fourier and cosine transforms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-33, October 1985, pp. 1341-1344.

[48] S. C. Chan and K. L. Ho, "Direct methods for computing discrete sinusoidal transforms", *IEE Proceedings: Radar and Signal Processing*, Vol. 137, Part F, December 1990, pp. 433–442.

[49] C. W. Kok, "Fast algorithm for computing discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 45, March 1997, pp. 757–760.

[50] Z. Wang, "Pruning the fast discrete cosine transform", *IEEE Transactions on Communications*, Vol. 39, No. 5, May 1991, pp. 640–643.

[51] Z. Jiang and A. N. Willson, "Fast cosine-modulated filter banks", *Proceedings of the International Conference on Digital Signal Processing (DSP'95)*, Limassol, Cyprus, June 1995, pp. 188–193.

[52] V. Britanak, "The fast DCT-IV/DST-IV computation via the MDCT", *Signal Processing*, Vol. 83, August 2003, pp. 1803–1813.

[53] S. Egner and M. Pueschel, "Automatic generation of fast discrete signal transforms", *IEEE Transactions on Signal Processing*, Vol. 49, No. 9, September 2001, pp. 1992–2002.

[54] M. Pueschel and J. M. Moura, "The algebraic approach to the discrete cosine and sine transforms and their fast algorithms", *SIAM Journal of Computing*, Vol. 32, No. 5, 2003, pp. 1280–1316.

[55] G. Plonka and M. Tasche, "Fast and numerically stable algorithms for discrete cosine transforms", *Linear Algebra and its Applications*, Vol. 394, No. 1, January 2005, pp. 309–345.

[56] E. Feig and S. Winograd, "On the multiplicative complexity of discrete cosine transforms", *IEEE Transactions on Information Theory*, Vol. 38, No. 4, July 1992, pp. 1387–1391.

[57] S. C. Chan and K. L. Ho, "A new two-dimensional fast cosine transform algorithm", *IEEE Transactions on Signal Processing*, Vol. 39, No. 2, February 1991, pp. 481–485.

[58] H. R. Wu and F. J. Paoloni, "A two-dimensional fast cosine transform algorithm", *Proceedings of International Conference on Image Processing*, Singapore, September 1989, pp. 50–54.

[59] H. R. Wu and F. J. Paoloni, "A structural approach to two-dimensional direct fast discrete cosine transform algorithms", *Proceedings of the International Symposium on Computer Architectures and DSP*, Hong Kong, October 1989.

[60] H. R. Wu and F. J. Paoloni, "A two-dimensional fast cosine transform algorithm based on Hou's approach", *IEEE Transactions on Signal Processing*, Vol. 39, No. 2, February 1991, pp. 544–546.

[61] V. Britanak and K. R. Rao, "Two-dimensional DCT/DST universal computational structure for $2^m \times 2^n$ block sizes", *IEEE Transactions on Signal Processing*, Vol. 48, No. 11, November 2000, pp. 3250–3255.

[62] E. Feig and S. Winograd, "Fast algorithms for the discrete cosine transform", *IEEE Transactions on Signal Processing*, Vol. 40, No. 9, September 1992, pp. 2174–2193.

[63] M. Vetterli, "Fast 2-D discrete cosine transform", *Proceedings of the IEEE ICASSP'85*, Tampa, FL, March 1985, pp. 1538–1541.

[64] P. Duhamel and C. Guillemot, "Polynomial transform computation of the 2-D DCT", *Proceedings of the IEEE ICASSP'90*, Albuquerque, NM, April 1990, pp. 1515–1518.

[65] J. Prado and P. Duhamel, "A polynomial transform-based computation of the 2-D DCT with minimum multiplicative complexity", *Proceedings of the IEEE ICASSP'96*, Atlanta, GA, May 1996, pp. 1347–1350.

[66] N. I. Cho, I. D. Yun and S. U. Lee, "A fast algorithm for 2-D DCT", *Proceedings of the IEEE ICASSP'91*, Toronto, Canada, May 1991, pp. 2197–2200.

[67] N. I. Cho and S. U. Lee, "A fast 4×4 DCT algorithm for the recursive 2-D DCT", *IEEE Transactions on Signal Processing*, Vol. 40, No. 9, September 1992, pp. 2166–2173.

[68] N. I. Cho and S. U. Lee, "Fast algorithm and implementation of 2-D discrete cosine transform", *IEEE Transactions on Circuits and Systems*, Vol. 38, No. 3, March 1991, pp. 297–305.

[69] N. I. Cho and S. U. Lee, "On the regular structure for the fast 2-D DCT algorithm", *IEEE Transactions on Circuits and Systems-II: Analog and Digital Signal Processing*, Vol. 40, No. 4, April 1993, pp. 259–266.

[70] Y. M. Huang, J. L. Wu and C. T. Hsu, "A refined fast 2-D discrete cosine transform algorithm with regular butterfly structure", *IEEE Transactions on Consumer Electronics*, Vol. 44, No. 2, May 1998, pp. 376–383.

[71] Y. M. Huang and J. L. Wu, "A refined fast 2-D discrete cosine transform algorithm", *IEEE Transactions on Signal Processing*, Vol. 47, No. 3, March 1999, pp. 904–907.

[72] H. R. Wu and Z. Man, "Comments on fast algorithm and implementation of 2-D discrete cosine transform", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 8, No. 2, April 1998, pp. 128–129.

CHAPTER 5

Integer Discrete Cosine/Sine Transforms

5.1 Introduction

The trend of low-powered discrete orthogonal sinusoidal transforms has become essential due to the explosion of wireless technologies, especially for applications which run on batteries. Digital image/video/audio signals are integer-valued which are quantized into M-bit representations. The discrete cosine transforms (DCTs) and discrete sine transforms (DSTs) are real-valued transforms that map integer-valued signals to floating-point coefficients. Although the fast algorithms for their computation reduce the number of arithmetic operations significantly, they still need floating-point operations. Another disadvantage associated with most of the fast algorithms that employ floating-point multiplications is the difficulty in applying them to lossless compression due to finite-length representation and corresponding round-off errors. More importantly, floating-point implementations in hardware are slow, require too much memory and consume too much power. In the field of VLSI, it is well known that floating-point multiplication is the operation consuming the most time and power and, therefore, causes the resulting devices to be large and expensive.

To achieve faster realization, floating-point multipliers in most practical implementations of discrete sinusoidal transforms are scaled up and approximated by integers. Specifically, the implementation is realized in fixed-point arithmetic where each floating-point multiplier (in internal representation: sign, exponent and mantissa) is approximated and derived in the form $\pm m \cdot 2^b$, where b is an integer exponent and m is an integer mantissa which can be a very large factor. This ad hoc approximation method still has problems: it does not reduce the computational complexity much and it introduces truncation errors. Therefore, it is important to develop new algorithms for the DCT and DST computation so that the dependency on floating-point multiplications can be reduced or completely eliminated.

The DCTs and DSTs with integer coefficients are of great interest since they lead to low-cost systems that can be designed more simply and implemented more efficiently with low-power consumption. The research in this direction has great potential for mobile computing and hand-held devices which run on batteries. The DCT and DST implementations based on integer approximation are currently modern transform technologies adaptable to available

power resources and suitable for transform-based lossless coding. The resulting integer transforms are comparable with the original real-valued transforms, preserve all their basic mathematical properties (linearity, orthogonality, symmetry of the basis vectors and recursivity) and performance such as transform coding gain or transform efficiency. The methods of integer approximation enable us to construct and flexibly generate a family of integer transforms with arbitrary accuracy and performance with the fast, efficient in-place multiplierless implementation using only binary additions and shifts. Thus, fast multiplierless integer transforms can replace the corresponding real-valued transforms in future wireless and satellite communications, as well as portable computing applications.

This chapter is devoted to integer DCT and DST. As the chapter is heavily manipulated with matrices, the first section presents the basic material from linear algebra, theory of matrices and matrix computations which is fundamental for understanding approximation methods (basic linear algebra and matrix decompositions are also discussed in Appendices A.1–A.3). In order to evaluate the approximation error between the approximated and original transform matrix, and to measure the performance of resulting approximated transform used in data compression applications, some theoretical criteria are defined in the second section. Finally, in the last three sections various developed methods and design approaches to integer approximation of DCTs and DSTs including the latest developments are described in detail. The chapter concludes with a summary, problems and exercises, and comprehensive references related to integer DCT and DST.

The chapter contains many new/improved/modified results. The fast integer DCTs and DSTs in the form of generalized signal flow graphs are ready to be used in practical applications.

5.2 Plane rotation matrices: factorizations and notations

Principally, many proposed fast algorithms for the efficient discrete trigonometric transforms computation define for a given DCT and DST matrices a sparse matrix factorization, in many cases consisting of butterfly matrices with elements ± 1, and cascade of plane rotations being the Givens–Jacobi rotations or Householder reflections [7–9].

One class of modern methods for integer approximation of DCTs and DSTs is based on the fact that Givens–Jacobi rotations and Householder reflections can be further factorized into a product of structurally simpler matrices, the so-called elementary matrices, defining an efficient computational structure called in engineering literature "the ladder network" [7] or "(scaled) lifting scheme" [8]. Consequently, having the sparse matrix factorization of the real-valued transform matrix and factorizations of Givens–Jacobi rotations and Householder reflections, any DCT and DST can be approximated and implemented by reversible integer-to-integer mapping or even they can be realized by multiplierless implementation in the form of binary adds and shifts only, which is very desirable for lossless transform coding.

Although the factorizations of Givens–Jacobi rotations and Householder reflections have been heavily discussed in literature [7–9], a little effort has been devoted to explain the origin of these factorizations exclusively with reference to basic results of linear algebra and classical theory of matrices. Indeed, using the basic material from linear algebra and

matrix theory, all the factorizations can be derived by more understandable way. Therefore, in the following sections the review of basic material from linear algebra and matrix theory (basic ideas and theorems) [1–5] is presented, which are indispensable to understand the material in this chapter related to integer approximation methods of DCTs and DSTs. We note that Appendices A.1 and A.2 deal also with basic linear algebra, specifically, vector spaces (some important concepts) and the matrix eigenvalue problem are discussed in detail. As well, Appendix A.3 covers matrix decompositions.

5.2.1 The determinant

If A is a square nonsingular matrix of order N then its determinant is given by $\det(A) = a$, $a \in R$. The determinant of A is defined in terms of $N-1$ determinants [5]:

$$\det(A) = \sum_{j=1}^{N} (-1)^{j+1} \det(A_{1j}), \tag{5.1}$$

where A_{1j} is an $(N-1) \times (N-1)$ matrix obtained by deleting the first row and jth column of A. Useful properties of the determinant include:

- $\det(AB) = \det(A)\det(B)$, A and B are nonsingular matrices,
- $\det(A^{\mathrm{T}}) = \det(A)$, T denotes matrix transposition,
- $\det(cA) = c^N \det(A)$, $c \in R$,
- $\det(A) \neq 0$ if A is nonsingular.

5.2.2 Orthogonal/orthonormal matrices

Matrix A is called orthogonal, if $AA^{\mathrm{T}} = I$, where I is the identity matrix. Additionally, if the norm of each row (basis vector) of the matrix is equal to 1, then the matrix is orthonormal. The orthogonal/orthonormal matrices possess a few useful properties [1]:

- The identity matrix I is orthogonal/orthonormal.
- If A is orthogonal/orthonormal, then $A^{-1} = A^{\mathrm{T}}$.
- If A is orthogonal/orthonormal, then A^{T} is also orthogonal/orthonormal.
- The product of two orthogonal/orthonormal matrices is an orthogonal/orthonormal matrix.
- Determinant of orthogonal/orthonormal matrix is equal to ± 1. If $\det(A) = +1$, then A is called to be eigenorthogonal/eigenorthonormal. Otherwise, if $\det(A) = -1$, then A is called to be non-eigenorthogonal/non-eigenorthonormal.

5.2.3 Triangular matrices and algebra of triangular matrices

A matrix with all elements under/above the main diagonal equal to zero is called an upper/lower triangular matrix. A unit triangular matrix is triangular matrix with 1s on the

main diagonal. There are a few useful properties about products, inverses and determinants of triangular matrices [5]:

- The inverse of upper (lower) triangular matrix is upper (lower) triangular.
- The inverse of unit upper (unit lower) triangular matrix is unit upper (unit lower) triangular.
- The product of two upper (lower) triangular matrices is upper (lower) triangular matrix.
- The product of two unit upper (unit lower) triangular matrices is unit upper (unit lower) triangular matrix.
- Determinant of upper or lower triangular matrix is equal to the product of its diagonal elements.
- Determinant of unit upper or unit lower triangular matrix is equal to 1.

5.2.4 Absolute value of a matrix and matrix/vector norms

The matrix/vector norms are frequently used for the analysis of matrix algorithms in linear algebra and matrix computations. They provide a measure of distance on the space of matrices/vector space, or more precisely, the space of matrices/vector space together with matrix/vector norms define a metric space [4, 5]. In general, matrix norms are defined for an arbitrary matrix, i.e., also for nonsquare matrices.

The absolute value (modulus) of a matrix $A = \{a_{ij}\}$ is the matrix [4]:

$$|A| = \{|a_{ij}|\}, \tag{5.2}$$

where $|a_{ij}|$ are moduli of the elements of A. If A and B are matrices of the same type for which the operations $A + B$ and $A \cdot B$ are defined, then

- $|A + B| \leq |A| + |B|$,
- $|A \cdot B| \leq |A| \cdot |B|$,
- $|\alpha \cdot A| = |\alpha| \cdot |A|, \quad \alpha \in R$,
- $|A^r| \leq |A|^r$ for a square matrix A, where $r > 0$ is an integer.

The norm of a matrix $A = \{a_{ij}\}$ is a real number $\|A\|$ satisfying the following properties [4, 5]:

- $\|A\| \geq 0, \quad (\|A\| = 0 \text{ iff } A = 0)$,
- $\|\alpha \cdot A\| = |\alpha| \cdot \|A\|, \quad \alpha \in R$, and in particular $\|-A\| = \|A\|$,
- $\|A + B\| \leq \|A\| + \|B\|$,
- $\|A \cdot B\| \leq \|A\| \cdot \|B\|$ and in particular $\|A^r\| \leq \|A\|^r$, where $r > 0$ is integer.

One important inequality between the norms of matrices A and B of the same type is given by:

$$\|A - B\| \geq |\,\|B\| - \|A\|\,|. \tag{5.3}$$

The matrix norm is called canonical, if satisfies additional properties:

- $|a_{ij}| \leq \|A\|$,
- The inequality $|A| \leq |B|$ implies that $\|A\| \leq \|B\|$.

Subscripts on $\|\cdot\|$ are used to distinguish between various norms. The most frequently used and easily computed matrix norms are [4, 5]:

$$\|A\|_{\mathrm{F}} = \sqrt{\sum_i \sum_j |a_{ij}|^2}, \quad \text{Frobenius norm,} \tag{5.4}$$

$$\|A\|_1 = \max_i \sum_j |a_{ij}|, \quad \text{1-norm,} \tag{5.5}$$

$$\|A\|_\infty = \max_j \sum_i |a_{ij}|, \quad \infty\text{-norm.} \tag{5.6}$$

It can be verified that matrix norms $\|A\|_{\mathrm{F}}$, $\|A\|_1$ and $\|A\|_\infty$ are canonical. For a vector $\mathbf{x} = [x_1, x_2, \ldots, x_N]^{\mathrm{T}}$ these norms are defined as

$$\|\mathbf{x}\|_2 = |\mathbf{x}| = \sqrt{|x_1|^2 + |x_2|^2 + \cdots + |x_N|^2}, \quad \text{Euclidean norm or 2-norm,} \tag{5.7}$$

$$\|\mathbf{x}\|_1 = |x_1| + |x_2| + \cdots + |x_N|, \tag{5.8}$$

$$\|\mathbf{x}\|_\infty = \max_i |x_i|, \quad \text{maximum norm.} \tag{5.9}$$

A unit vector with respect to the norm $\|\cdot\|$ is a vector $\mathbf{x}$ that satisfies $\|\mathbf{x}\| = 1$. We note that vector norms are derived from the class of vector p-norms defined as [5]

$$\|\mathbf{x}\|_p = (|x_1|^p + |x_2|^p + \cdots + |x_N|^p)^{\frac{1}{p}}, \quad p \geq 1.$$

A very important property concerning the vector norms is Cauchy–Schwartz inequality:

$$|\mathbf{x}\mathbf{y}^{\mathrm{T}}| \leq \|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2. \tag{5.10}$$

Finally, for the identity matrix of order N we have

$$\|I\|_{\mathrm{F}} = \sqrt{N}, \quad \|I\|_1 = \|I\|_\infty = 1.$$

5.2.5 Elementary rotation matrices

Elementary rotation matrices G_{ij} are defined as [1]:

$$G_{ij} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & -s & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad G_{ij}^{-1} = G_{ij}^{\mathrm{T}} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & -s & \dots & c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \tag{5.11}$$

where $c^2 + s^2 = 1$, $c = \cos\varphi$, $s = \sin\varphi$ for some angle φ, and elementary rotation matrices H_{ij} are defined as [5]:

$$H_{ij} = \begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & c & \dots & s & & \\ & & \vdots & \ddots & \vdots & & \\ & & s & \dots & -c & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \quad H_{ij}^{-1} = H_{ij}^{\mathrm{T}} = H_{ij}. \tag{5.12}$$

Subscript indices i and j ($i < j$) indicate positions of elements c and s in the matrix. The elementary rotation matrices given by (5.11) and (5.12) differ from the identity matrix only by four elements placed on crossing two rows and two columns. They are orthogonal transformations playing a central role in the least squares solutions of overdetermined systems of linear equations and symmetric eigenvalue problems [5]. Elementary rotation matrices G_{ij} are known as the Givens–Jacobi rotations. They are eigenorthogonal, i.e., $\det(G_{ij}) = \det(G_{ij}^{\mathrm{T}}) = +1$, whereby G_{ij} and G_{ij}^{T} are inverses to each other. On the other hand, elementary rotation matrices H_{ij} are known as the Householder reflections. They are non-eigenorthogonal, i.e., $\det(H_{ij}) = \det(H_{ij}^{-1}) = -1$, whereby $H_{ij}^{-1} = H_{ij}^{\mathrm{T}} = H_{ij}$. Through the premultiplications (multiplications on the left) and/or postmultiplications (multiplications on the right) of a nonsingular matrix by elementary rotation matrices we can reduce the given matrix into various canonical forms introducing zero elements in the matrix by properly choosing the rotation angle or reflection plane. For our purposes it will be sufficient to consider the simplest case of 2×2 elementary rotation matrices.

A 2×2 orthogonal (eigenorthogonal) matrix is called Givens–Jacobi rotation, if it has the form [5]:

$$G_{\varphi} = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad G_{\varphi}^{-1} = G_{\varphi}^{\mathrm{T}} = G_{-\varphi}. \tag{5.13}$$

A 2×2 orthogonal (non-eigenorthogonal) matrix is called Householder reflection, if it has the form [5]:

$$H_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & -\cos\varphi \end{pmatrix}, \quad H_\varphi^{-1} = H_\varphi^{\mathrm{T}} = H_\varphi. \tag{5.14}$$

Note: All coefficients in fast Fourier transform (FFT) algorithms are complex numbers with the magnitude equal to 1, i.e., every coefficient can be expressed in the form [48, 49]:

$$\mathrm{e}^{\mathrm{j}\theta} = \cos\theta + \mathrm{j}\sin\theta = c + \mathrm{j}s,$$

where $\mathrm{j} = \sqrt{-1}$. If we denote $a = c + \mathrm{j}s$, where $c^2 + s^2 = 1$ and $|a| = 1$, then the complex multiplication $\mathbf{y} = a\mathbf{x}$, where $x = x_r + \mathrm{j}x_i$, can be written as

$$\mathbf{y} = a\mathbf{x} = (c + \mathrm{j}s)(x_r + \mathrm{j}x_i) = (cx_r - sx_i) + \mathrm{j}(sx_r + cx_i),$$

or in the equivalent matrix–vector notation as

$$\mathbf{y} = (1 \quad \mathrm{j}) \begin{pmatrix} c & -s \\ s & c \end{pmatrix} \begin{pmatrix} x_r \\ x_i \end{pmatrix}, \quad s \neq 0. \tag{5.15}$$

The matrix on the right-hand side of (5.15) is the 2×2 Givens–Jacobi rotation.

5.2.6 Elementary transformations

Elementary transformations are elementary matrices of a special form frequently performed upon matrices as follows:

1. Multiplication to the elements of some row by a number α.
2. Adding to the elements of some row numbers proportional (α multiple) to the elements of some preceding row.
3. Adding to the elements of some row numbers proportional (α multiple) to the elements of some following row.

Sometimes such elementary transformations are made upon the columns of the matrix. Any elementary transformation of the rows is equivalent to a premultiplication of the

matrix by a nonsingular matrix of a special form. Thus, the operation (1) is equivalent to a premultiplication by the matrix

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & & \alpha & & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}. \tag{5.16}$$

Operation (2) is equivalent to a premultiplication by the matrix

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & & & & \\ & & \vdots & \ddots & & & \\ & & \alpha & \dots & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}, \tag{5.17}$$

and the operation (3) is equivalent to a premultiplication by the matrix

$$\begin{pmatrix} 1 & & & & & & \\ & \ddots & & & & & \\ & & 1 & \dots & \alpha & & \\ & & & \ddots & \vdots & & \\ & & & & 1 & & \\ & & & & & \ddots & \\ & & & & & & 1 \end{pmatrix}. \tag{5.18}$$

Operations (1)–(3) can be performed alternatively upon the columns in postmultiplications of the matrix.

Note: In general, the elementary rotation matrices can be thought of as the elementary transformations, which are orthogonal.

5.2.7 QR, LU, LDU and PLUS factorizations

We mentioned that in linear algebra and matrix computations, a real nonsingular matrix is reduced by elementary rotation matrices and elementary transformations into various canonical forms in order to simplify subsequent computational steps of a solved problem.

Such procedures lead to various useful factorizations of the matrix into the products of structurally simpler matrices.

There exist two basic methods how to reduce a real nonsingular matrix of order N into equivalent upper triangular form. The first method is based on premultiplications of the matrix by elementary Givens–Jacobi rotation matrices. This procedure leads to the well-known QR factorization, where Q is an orthogonal matrix and R is an upper triangular matrix. The QR factorization is also discussed in the Appendix A.3 (see equations (A.27)–(A.34)). The following theorem and corollaries state the QR factorization [1].

Theorem 5.1: ([1], Chapter 1, p. 37)
Arbitrary real nonsingular matrix

$$A = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} \end{pmatrix}$$

can be reduced through successive premultiplications by elementary Givens–Jacobi rotation matrices G_{ij} to an upper triangular matrix, whose all diagonal elements are positive besides the last one, i.e.,

$$A^{(n-1)} = G_{n-1,n} \dots G_{13}\, G_{12}\, A = \begin{pmatrix} a_{11}^{(1)} & a_{12}^{(1)} & \dots & a_{1n}^{(1)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn}^{(n-1)} \end{pmatrix}.$$

•

Corollary 5.1: (QR factorization)
Arbitrary real nonsingular matrix A is the product of an eigenorthogonal matrix Q and an upper triangular matrix R, i.e., $A = QR$, where

$$Q = (G_{n-1,n} \cdots G_{13}\, G_{12})^{-1} \quad \text{and} \quad R = A^{(n-1)}.$$

•

Corollary 5.2:
Any eigenorthogonal matrix A is the product of at most $\frac{n(n-1)}{2}$ elementary Givens–Jacobi rotation matrices, i.e.,

$$A = G_{12}^{-1}\, G_{13}^{-1} \cdots G_{n-1,n}^{-1} A^{(n-1)}, \quad \text{where} \quad A^{(n-1)} = I.$$

•

The second method how to reduce a real nonsingular matrix to the upper triangular form is based on premultiplications of the matrix by elementary transformations, specifically, by elementary matrices of the form (2). The procedure leads to the well-known LU factorization, where L is a lower triangular matrix and U is an upper triangular matrix. The LU factorization is also discussed in the Appendix A.3 (see equations (A.18)–(A.24)). The following theorem states LU factorization [1].

Theorem 5.2: ([1], Chapter 1, p. 20, and its general form in Ref. [3], Chapter 2, p. 50) On condition that the principal minors of the matrix

$$A = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix}$$

are not equal to zero, i.e., that $a_{11} \neq 0, \dots, \det(A) \neq 0$, the matrix A may be represented as the product of a lower triangular matrix and an upper triangular matrix.

•

Note: LU factorization of a matrix A will be uniquely determined if we prescribe the values for the diagonal elements of one of the triangular matrices. It is convenient to consider, for example, that elements of U are equal to $u_{ii} = 1, i = 1, 2, \dots, N$ [1, 3].

LU factorization is actually originated from the method of Gaussian elimination used for solving systems of linear equations [3]. One step of Gaussian elimination is equivalent to a premultiplication of the matrix by elementary matrix of the form (2) which is the unit lower triangular and it is called Gauss elementary matrix. Thus, the transition from original matrix A to its upper triangular form can be written as

$$W_m W_{m-1} \dots W_1 A = U,$$

where W_i, $i = 1, 2, \dots, m$ are Gauss elementary matrices. Then, we have

$$A = (W_m W_{m-1} \dots W_1)^{-1} U = LU.$$

From the algebra of triangular matrices it follows that matrix L is the unit lower triangular matrix. Moreover, if $\det(A) = +1$, then $\det(LU) = \det(L) \det(U) = +1$, then $\det(L) = +1$ and $\det(U) = +1$. Generally, if we take into account interchanges of two rows in the matrix during the factorization process (the so-called pivoting operation in Gaussian elimination), then we should consider in LU factorization premultiplied or postmultiplied permutation matrices P_i.

In addition, according to the theorem ([3], Chapter 2, p. 53), an arbitrary matrix A, whose all principal minors are not equal to zero, can be represented as the product of $A = LDU$, where L is an unit lower triangular matrix, D is a diagonal matrix and U is an unit upper triangular matrix.

A variant of Gaussian elimination is Jordan elimination. Whereas the Gaussian elimination leads to an upper triangular matrix and the Jordan elimination leads to a diagonal matrix. One step of Jordan elimination is equivalent to a premultiplication of the matrix by

$$\begin{pmatrix} 1 & 0 & \dots & \alpha_{1,i} & \dots & 0 \\ 0 & 1 & \dots & \alpha_{2,i} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & & \vdots \\ 0 & 0 & \dots & \alpha_{i-1,i} & \dots & 0 \\ 0 & 0 & \dots & 1 & \dots & 0 \\ 0 & 0 & \dots & \alpha_{i+1,i} & \dots & 0 \\ \vdots & \vdots & & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \alpha_{n,i} & \dots & 1 \end{pmatrix}, \tag{5.19}$$

called the Jordan matrix. It can be easily verified that Jordan matrix is the product of elementary matrices of the forms (2) and (3) (see elementary transformations).

The *LU* factorization is the basic approach to factorize an invertible matrix into the product of triangular matrices and possibly permutation matrices taking into account row interchanges during the factorization process. In the special case, if a real nonsingular matrix A has its determinant equal to $+1$, i.e., $\det(A) = +1$, then it can be formulated theorem for general *PLUS* matrix factorization of A [9–11] as follows.

Theorem 5.3: (*PLUS* factorization)
A real square nonsingular matrix A has a factorization of $A = PLUS$ if and only if $\det(A) = \det(P) = +1$, where P is a permutation matrix, L is an unit lower triangular, U is an unit upper triangular and S is an unit lower triangular matrix of the form:

$$S = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ s_1 & s_2 & \dots & s_{n-1} & 1 \end{pmatrix}, \quad \text{where } S^{-1} = \begin{pmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & 1 & 0 \\ -s_1 & -s_2 & \dots & -s_{n-1} & 1 \end{pmatrix}.$$

•

Note: In general, any real square nonsingular matrix A can be customizably factorized into three triangular matrices, $A = PLUS$ customizable factorization, where P is a permutation matrix (in some cases P may be the identity matrix), U is an upper triangular matrix of which the diagonal elements $d_1, d_2, \dots, d_N$ are customizable and they can be given by all means as long as its determinant is equal to that of A up to a possible sign adjustment, i.e., $\det(A) = \det(U) = d_1\, d_2 \dots d_N$. S is a unit lower or upper triangular matrix of which all but $N - 1$ off-diagonal elements are also flexibly customizable such as a single-row, single-column, bidiagonal matrix or other specially patterned matrices. Besides *PLUS*,

a customizable factorization also has other alternatives, *LUSP*, *PSUL* and *SULP* for unit lower triangular S, and *PULS*, *ULSP*, *PSLU*, *SLUP* for unit upper triangular S [10].

We note that any nonsingular matrix A with $\det(A) = -1$ can be scaled to have its $\det(A) = +1$. More insight into the structure of *PLUS* factorization gives its factorization algorithm [9, 10]. Let A be a real nonsingular (invertible) matrix of order N:

$$A = \begin{pmatrix} a_{11}^{(0)} & a_{12}^{(0)} & \dots & a_{1n}^{(0)} \\ a_{21}^{(0)} & a_{22}^{(0)} & \dots & a_{2n}^{(0)} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^{(0)} & a_{n2}^{(0)} & \dots & a_{nn}^{(0)} \end{pmatrix}.$$

Then, there must exist a permutation matrix P_1 for row interchanges such that

$$P_1 A = \begin{pmatrix} p_{11}^{(1)} & p_{12}^{(1)} & \cdots & p_{1n}^{(1)} \\ p_{21}^{(1)} & p_{22}^{(1)} & \cdots & p_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(1)} & p_{n2}^{(1)} & \cdots & p_{nn}^{(1)} \end{pmatrix},$$

and $p_{1n}^{(1)} \neq 0$, and hence there must exist a number s_1 such that $p_{11}^{(1)} - s_1 p_{1n}^{(1)} = 1$. Then, we get $s_1 = \frac{p_{11}^{(1)} - 1}{p_{1n}^{(1)}}$ and we obtain a product of

$$P_1 A S_{0,1} = P_1 A \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ -s_1 & 0 & \dots & 1 \end{pmatrix} = \begin{pmatrix} 1 & p_{12}^{(1)} & \dots & p_{1n}^{(1)} \\ p_{21}^{(1)} - s_1 p_{2n}^{(1)} & p_{22}^{(1)} & \dots & p_{2n}^{(1)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1}^{(1)} - s_1 p_{nn}^{(1)} & p_{n2}^{(1)} & \dots & p_{nn}^{(1)} \end{pmatrix}.$$

The second step is Gaussian elimination of the first column and it is achieved by the premultiplication of product $P_1 A S_{0,1}$ by Gauss elementary matrix L_1 as follows:

$$L_1 P_1 A S_{0,1} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ s_1 p_{2n}^{(1)} - p_{21}^{(1)} & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ s_1 p_{nn}^{(1)} - p_{n1}^{(1)} & 0 & \dots & 1 \end{pmatrix} P_1 A S_{0,1} = \begin{pmatrix} 1 & a_{12}^{(2)} & \dots & a_{1n}^{(2)} \\ 0 & a_{22}^{(2)} & \dots & a_{2n}^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & a_{n2}^{(2)} & \dots & a_{nn}^{(2)} \end{pmatrix}.$$

Continuing the factorization process for $k = 2, 3, \dots, N-1$, where P_k defines the row interchanges among the kth through Nth rows to guarantee that kth element in the Nth column are not equal to zero, i.e., $p_{kn}^{(k)} \neq 0$, matrices $S_{0,k}$ convert elements $a_{kk}^{(k)}$ into $1s$,

where $s_k = \frac{p_{kk}^{(k)} - 1}{p_{kn}^{(k)}}$, and matrices L_k represent row multipliers used for Gaussian elimination of column k. Completing the factorization process we get the product:

$$L_{N-1}\, P_{N-1} \ldots L_2\, P_2\, L_1\, P_1\, A\, S_{0,1}\, S_{0,3} \ldots S_{0,N-1} = \begin{pmatrix} 1 & a_{12}^{(N-1)} & \ldots & a_{1n}^{(N-1)} \\ 0 & 1 & \ldots & a_{2n}^{(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{nn}^{(N-1)} \end{pmatrix} = U.$$

Since $\det(A) = +1$, $a_{nn}^{(N-1)} = 1$ and U is the unit upper triangular. Having, respectively, multiplied all matrices $S_{0,k}$ together all permutation matrices P_k together and all unit lower triangular matrices L_k together, we have one matrix S^{-1}, one premultiplying matrix P^{T} and one unit lower triangular matrix L^{-1}. From algebra of triangular matrices it follows that the inverse of a unit lower triangular matrix is also a unit lower triangular matrix and we have

$$S_{0,1}\, S_{0,3} \ldots S_{0,N-1} = \begin{pmatrix} 1 & 0 & \ldots & 0 & 0 \\ 0 & 1 & \ldots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \ldots & 1 & 0 \\ -s_1 & -s_2 & \ldots & -s_{n-1} & 1 \end{pmatrix} = S^{-1}$$

and

$$\begin{aligned} &L_{N-1}\, P_{N-1} \ldots L_2\, P_2\, L_1\, P_1 \\ &\quad = L_{N-1}\, (P_{N-1}\, L_{N-2}\, P_{N-1}^{\mathrm{T}}) \ldots (P_{N-1} \ldots P_2\, L_1\, P_2^{\mathrm{T}} \ldots P_{N-1}^{\mathrm{T}})\, (P_{N-1} \ldots P_2\, P_1) \\ &\quad = L^{-1}\, P^{\mathrm{T}}, \end{aligned}$$

where

$$L^{-1} = L_{N-1}\, (P_{N-1}\, L_{N-2}\, P_{N-1}^{\mathrm{T}}) \ldots (P_{N-1} \ldots P_2\, L_1\, P_2^{\mathrm{T}} \ldots P_{N-1}^{\mathrm{T}})$$

and

$$P^{-1} = P^{\mathrm{T}} = P_{N-1} \ldots P_2\, P_1.$$

Hence, finally we obtain

$$L^{-1}\, P^{-1}\, A\, S^{-1} = U \quad \text{or} \quad A = PLUS.$$

5.2.8 Matrix factorizations of Givens–Jacobi rotations and Householder reflections

Givens–Jacobi rotations and Householder reflections can be further factorized into the products of elementary matrices being the unit lower and unit upper triangular matrices

and possibly a diagonal matrix. In this section it is shown that the basic material from linear algebra and matrix theory is sufficient to derive all the factorizations of Givens–Jacobi rotations and Householder reflections. We will concentrate only on the simplest case of 2×2 matrices because of they are important in practical applications. A general matrix factorization theory to factorize an invertible matrix of higher order into the product of triangular matrices can be found in Refs. [9, 10].

Let G_φ and H_φ are respectively 2×2 Givens–Jacobi rotations and Householder reflections given by (5.13) and (5.14). Since $\det(G_\varphi) = +1$ according to *PLUS* factorization theorem we can apply to Givens–Jacobi rotations G_φ the *PLUS* factorization algorithm. The permutation matrix P in this case corresponds to the identity matrix and we can omit it. In the first step, there must exist a number s_1 such that $\cos\varphi + s_1 \sin\varphi = 1$ and hence $s_1 = \frac{1-\cos\varphi}{\sin\varphi}$ and we obtain the product of

$$G_\varphi S^{-1} = G_\varphi \begin{pmatrix} 1 & 0 \\ \frac{\cos\varphi - 1}{\sin\varphi} & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\sin\varphi \\ \frac{1-\cos\varphi}{\sin\varphi} & \cos\varphi \end{pmatrix}.$$

The second step is Gaussian elimination of the first column of $G_\varphi S^{-1}$, what is equivalent to the premultiplication of $G_\varphi S^{-1}$ by Gauss elementary matrix L^{-1} as follows:

$$L^{-1} G_\varphi S^{-1} = \begin{pmatrix} 1 & 0 \\ \frac{\cos\varphi - 1}{\sin\varphi} & 1 \end{pmatrix} G_\varphi \begin{pmatrix} 1 & 0 \\ \frac{\cos\varphi - 1}{\sin\varphi} & 1 \end{pmatrix} = \begin{pmatrix} 1 & -\sin\varphi \\ 0 & 1 \end{pmatrix} = U,$$

and we obtained the factorization of $L^{-1} G_\varphi S^{-1}$. From algebra of unit triangular matrices it follows that the inverse of unit lower/unit upper triangular matrix is also unit lower/unit upper triangular matrix, and we easily find their inverses as

$$\begin{pmatrix} 1 & 0 \\ x & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & 0 \\ -x & 1 \end{pmatrix}, \quad \begin{pmatrix} 1 & x \\ 0 & 1 \end{pmatrix}^{-1} = \begin{pmatrix} 1 & -x \\ 0 & 1 \end{pmatrix}.$$

Thus, we finally get the factorization of $G_\varphi = \mathrm{LUS}$ (and hence LUL factorization) given by

$$G_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix}, \quad \text{where } \sin\varphi \neq 0 \tag{5.20}$$

and

$$G_\varphi^{-1} = \begin{pmatrix} 1 & 0 \\ -\frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix}. \tag{5.21}$$

We note that $\tan\frac{\varphi}{2} = \frac{1-\cos\varphi}{\sin\varphi}$. The *PLUS* factorization of Givens–Jacobi rotations is the well known in engineering literature as the "ladder network" [7] or "lifting scheme" [8]. Factored matrices on the right-hand sides of (5.20) and (5.21) in engineering literature called "lifting matrices" are actually Gauss–Jordan elementary matrices being the unit

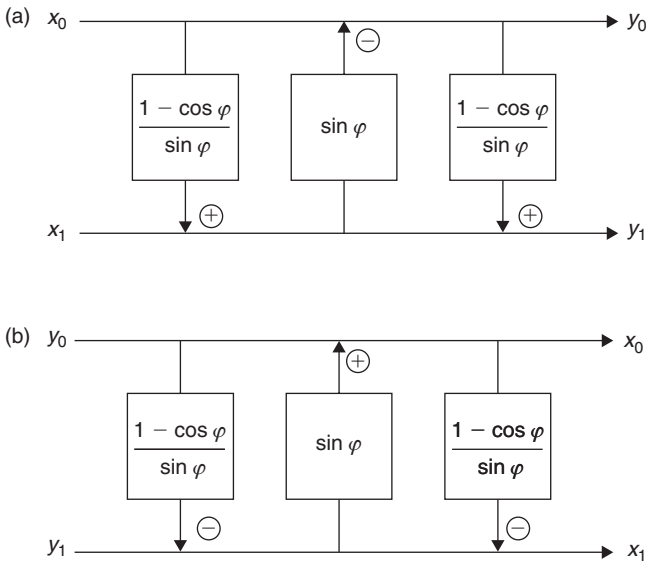


Fig. 5.1. The LUL structures for implementation of Givens–Jacobi rotations based on *PLUS* factorization: (a) forward and (b) inverse.

lower and unit upper triangular matrices. On the other hand, although the matrix G_φ is the orthogonal (eigenorthogonal), the factored matrices are not longer orthogonal, however they are invertible. The *PLUS* factorization of Givens–Jacobi rotations G_φ into the product of Gauss–Jordan elementary matrices defines computational structures for the efficient implementation of $\mathbf{y} = G_\varphi\,\mathbf{x}$ and $\mathbf{x} = G_\varphi^{-1}\,\mathbf{y}$. The corresponding LUL structures are respectively shown in Fig. 5.1(a) and (b), where $(x_0, x_1)^{\mathrm{T}}$ is the input data vector and $(y_0, y_1)^{\mathrm{T}}$ is rotated vector.

Since $G_\varphi = G_{-\varphi}^{\mathrm{T}}$, from (5.20) we obtain the alternative ULU factorization as

$$G_\varphi = \begin{pmatrix} 1 & -\frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \tag{5.22}$$

and

$$G_\varphi^{-1} = \begin{pmatrix} 1 & \frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1-\cos\varphi}{\sin\varphi} \\ 0 & 1 \end{pmatrix}. \tag{5.23}$$

The corresponding forward and inverse ULU structures are respectively shown in Fig. 5.2 (a) and (b).

From the implementation point of view, to invert Givens–Jacobi rotation given by *PLUS* factorization, we simply need to subtract out what was added in the forward computation.

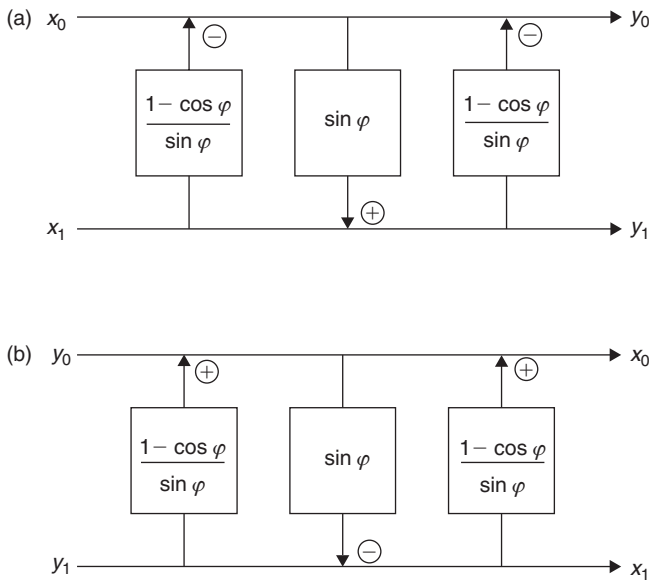


Fig. 5.2. The alternative ULU structures for implementation of Givens–Jacobi rotations based on *PLUS* factorization: (a) forward and (b) inverse.

The implementation of Givens–Jacobi rotation G_φ given by *PLUS* factorization compared to its direct implementation has the following advantages:

1. The number of multiplications is reduced from four to three multiplications and overall arithmetic complexity is 3 multiplications and 3 additions.
2. Leads to in-place implementation, i.e., without the need of auxilliary memory, which is the desired property in VLSI implementations.
3. Multipliers in factor matrices can be quantized (using functions such as *round*, *floor* or *ceil*) to obtain integer-to-integer mapping.
4. The perfect reconstruction property.

Note: The *PLUS* factorization theorem [9] has been originally formulated for a nonsingular matrix, whose determinant is equal to ± 1. In general, the *PLUS* factorization algorithm can be applied also to the Householder reflections H_φ, however one of factor matrices will have one element on main diagonal equal to -1. This is not correct with respect to the definition of the unit lower/unit upper triangular matrix. As an example, the *PLUS* factorization of H_φ is given by

$$H_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi \\ 0 & -1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1-\cos\varphi}{\sin\varphi} & 1 \end{pmatrix}, \quad \text{where } \sin\varphi \neq 0,$$

and in the corresponding computational structure we need to change the sign of one vector component during the computation.

In order to derive another factorizations of Givens–Jacobi rotations G_φ and Householder reflections H_φ we use the elementary transformations of the forms (2) and (3) (Gauss–Jordan elementary matrices) or Jordan elimination. At first, let us consider the Givens–Jacobi rotations given by equation (5.13). Adding $-(\frac{\sin\varphi}{\cos\varphi})$ multiple of the first row of G_φ to the second row is equivalent to the premultiplication of G_φ by Gauss–Jordan elementary matrix W_1 and we obtain the product:

$$W_1\, G_\varphi = \begin{pmatrix} 1 & 0 \\ -\frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} G_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix}.$$

Subsequently, adding $(\sin\varphi\cos\varphi)$ multiple of the second row of $W_1 G_\varphi$ to the first row is equivalent to the premultiplication of $W_1 G_\varphi$ by Gauss–Jordan elementary matrix W_2 and we get the product $W_2 W_1 G_\varphi = D$, where D is a diagonal matrix or

$$\begin{pmatrix} 1 & \sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} G_\varphi = \begin{pmatrix} \cos\varphi & 0 \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix} = D.$$

Using the algebra of unit triangular matrices (inverses) we obtain the factorization of $G_\varphi = W_1^{-1} W_2^{-1} D$ or

$$G_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix}, \quad \text{where } \cos\varphi \neq 0. \tag{5.24}$$

Since $G_{-\varphi}^{\mathrm{T}} = G_\varphi$, from (5.24) we finally get the DLU factorization

$$G_\varphi = \begin{pmatrix} \cos\varphi & 0 \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi\cos\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \tag{5.25}$$

and ULD factorization of G_φ^{-1}

$$G_\varphi^{-1} = \begin{pmatrix} 1 & \frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi\cos\varphi & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\cos\varphi} & 0 \\ 0 & \cos\varphi \end{pmatrix}, \tag{5.26}$$

which is the well known in engineering literature as "scaled lifting scheme" [7, 8]. The corresponding DLU and ULD structures are respectively shown in Fig. 5.3(a) and (b). We note that the DLU factorization can be directly derived from the *LU* factorization theorem ([3], Chapter 2, p. 50).

Since $G_{-\varphi}^{-1} = G_\varphi$, from (5.24) we obtain the alternative DUL factorization as

$$G_\varphi = \begin{pmatrix} \frac{1}{\cos\varphi} & 0 \\ 0 & \cos\varphi \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \tag{5.27}$$

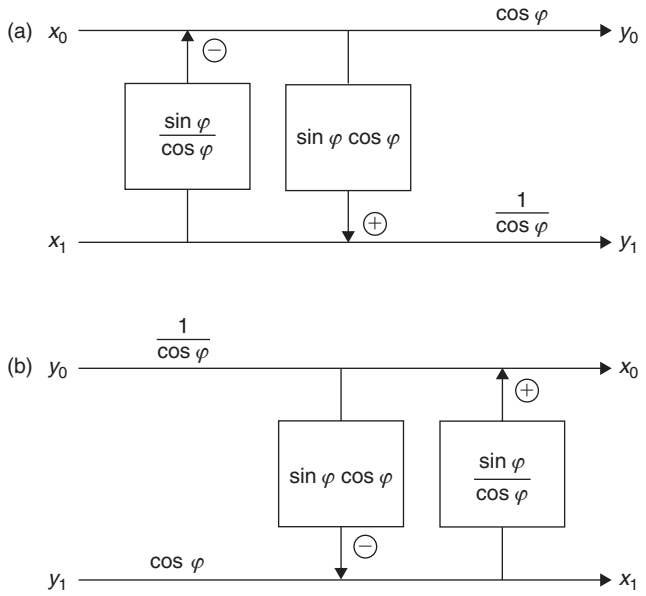


Fig. 5.3. The DLU and ULD structures for implementation of Givens–Jacobi rotations based on *LU* factorization.

and LUD factorization of G_φ^{-1}

$$G_\varphi^{-1} = \begin{pmatrix} 1 & 0 \\ -\frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix}. \tag{5.28}$$

The corresponding DUL and LUD structures are respectively shown in Fig. 5.4(a) and (b).

Now let us consider the Householder reflections H_φ. By the exactly same factorization procedure as for G_φ we get the factorization of H_φ as

$$H_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & -\frac{1}{\cos\varphi} \end{pmatrix}, \quad \text{where } \cos\varphi \neq 0. \tag{5.29}$$

Since $H_\varphi^{-1} = H_\varphi$, from (5.29) we get the DUL factorization

$$H_\varphi = \begin{pmatrix} \frac{1}{\cos\varphi} & 0 \\ 0 & -\cos\varphi \end{pmatrix} \begin{pmatrix} 1 & \sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \tag{5.30}$$

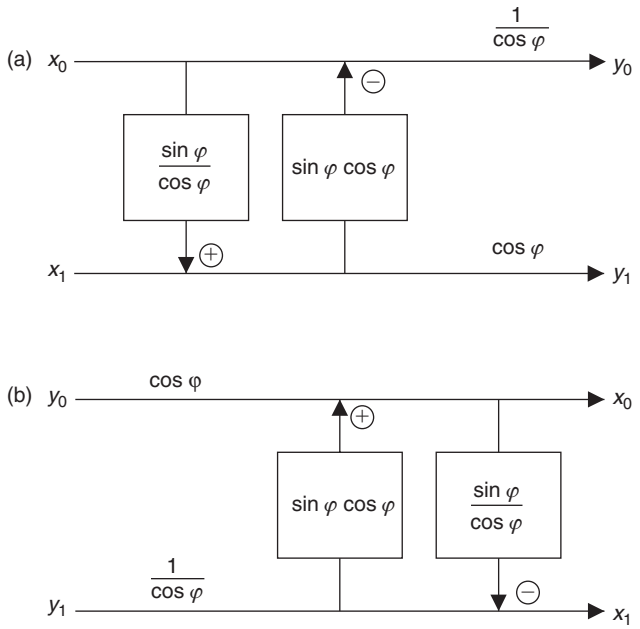


Fig. 5.4. The alternative DUL and LUD structures for implementation of Givens–Jacobi rotations based on *LU* factorization.

and LUD factorization of H_φ^{-1}

$$H_\varphi^{-1} = \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} 1 & -\sin\varphi\cos\varphi \\ 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & -\frac{1}{\cos\varphi} \end{pmatrix}. \tag{5.31}$$

The corresponding DUL structure is similar to that of G_φ except for sign changes.

Since $H_\varphi = H_\varphi^{\mathrm{T}}$, from (5.29) we obtain the alternative DLU factorization as

$$H_\varphi = \begin{pmatrix} \cos\varphi & 0 \\ 0 & -\frac{1}{\cos\varphi} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi\cos\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \tag{5.32}$$

and ULD factorization of H_φ^{-1}

$$H_\varphi^{-1} = \begin{pmatrix} 1 & -\frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi\cos\varphi & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\cos\varphi} & 0 \\ 0 & -\cos\varphi \end{pmatrix}. \tag{5.33}$$

For completeness, according to *LDU* factorization theorem ([3], Chapter 2, p. 49) we directly obtain for the Givens–Jacobi rotations and Householder reflections the

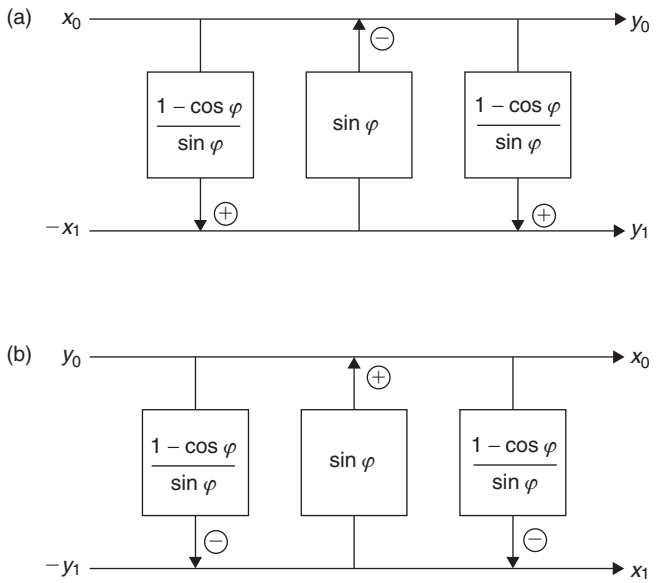


Fig. 5.5. The LUL structures for Givens–Jacobi rotations modified for the implementation of Householder reflections: (a) forward and (b) inverse.

following factorizations:

$$G_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & \frac{1}{\cos\varphi} \end{pmatrix} \begin{pmatrix} 1 & -\frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix}, \tag{5.34}$$

$$H_\varphi = \begin{pmatrix} 1 & 0 \\ \frac{\sin\varphi}{\cos\varphi} & 1 \end{pmatrix} \begin{pmatrix} \cos\varphi & 0 \\ 0 & -\frac{1}{\cos\varphi} \end{pmatrix} \begin{pmatrix} 1 & \frac{\sin\varphi}{\cos\varphi} \\ 0 & 1 \end{pmatrix}. \tag{5.35}$$

It is important to note that between the Givens–Jacobi rotations G_φ and Householder reflections H_φ the following relations hold:

$$G_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi \\ \sin\varphi & -\cos\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = H_\varphi \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}, \tag{5.36}$$

$$H_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} = G_\varphi \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{5.37}$$

It means that the computational structures for Givens–Jacobi rotations can be used by minor modification for the implementation of Householder reflections and vice versa. As an example, in Fig. 5.5 is shown the LUL structure for Givens–Jacobi rotations based on *PLUS* factorization modified for the implementation of Householder reflections.

In general, any other plane rotations occurring in practical applications such as

$$\bar{R}_\varphi = \begin{pmatrix} -\cos\varphi & \sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix}, \quad \bar{R}_\varphi^{-1} = \bar{R}_\varphi,$$

$$\tilde{R}_\varphi = \begin{pmatrix} \sin\varphi & \cos\varphi \\ \cos\varphi & -\sin\varphi \end{pmatrix}, \quad \tilde{R}_\varphi^{-1} = \tilde{R}_\varphi,$$

$$\hat{R}_\varphi = \begin{pmatrix} \sin\varphi & \cos\varphi \\ -\cos\varphi & \sin\varphi \end{pmatrix}, \quad \hat{R}_\varphi^{-1} = \hat{R}_\varphi^{\mathrm{T}},$$

and

$$\check{R}_\varphi = \begin{pmatrix} -\sin\varphi & \cos\varphi \\ \cos\varphi & \sin\varphi \end{pmatrix}, \quad \check{R}_\varphi^{-1} = \check{R}_\varphi,$$

can be respectively converted to the Givens–Jacobi rotations as follows:

$$\bar{R}_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix} = G_{-\varphi} \begin{pmatrix} -1 & 0 \\ 0 & 1 \end{pmatrix},$$

$$\tilde{R}_\varphi = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = G_{-\varphi} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix},$$

$$\hat{R}_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} = G_{\varphi} \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix},$$

and

$$\check{R}_\varphi = \begin{pmatrix} \cos\varphi & -\sin\varphi \\ \sin\varphi & \cos\varphi \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} = G_{\varphi} \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

5.2.9 Evaluating the determinants of DCT/DST matrices

Exploiting the properties of determinant and relations between DCT and DST matrices we can evaluate the determinants of explicit forms of orthonormal DCT/DST matrices presented in Section 4.2 for $N = 2, 4$ and 8 as follows:

$$\begin{aligned} &\det(C_3^{\mathrm{I}}) = -1, \ \det(S_3^{\mathrm{I}}) = -1, \ \det(\tilde{C}_2^{\mathrm{I}}) = -1, \ \det(\tilde{S}_2^{\mathrm{I}}) = -1, \\ &\det(C_5^{\mathrm{I}}) = +1, \ \det(S_7^{\mathrm{I}}) = -1, \ \det(\tilde{C}_4^{\mathrm{I}}) = +1, \ \det(\tilde{S}_4^{\mathrm{I}}) = +1, \\ &\det(C_9^{\mathrm{I}}) = +1, \qquad\qquad\quad \det(\tilde{C}_8^{\mathrm{I}}) = +1, \ \det(\tilde{S}_8^{\mathrm{I}}) = +1, \end{aligned}$$

and

$$\det(C_2^{\mathrm{II}}) = \det(C_2^{\mathrm{III}}) = -1, \quad \det(S_2^{\mathrm{II}}) = \det(S_2^{\mathrm{III}}) = -1, \quad \det(C_2^{\mathrm{IV}}) = \det(S_2^{\mathrm{IV}}) = -1,$$
$$\det(C_4^{\mathrm{II}}) = \det(C_4^{\mathrm{III}}) = +1, \quad \det(S_4^{\mathrm{II}}) = \det(S_4^{\mathrm{III}}) = +1, \quad \det(C_4^{\mathrm{IV}}) = \det(S_4^{\mathrm{IV}}) = +1,$$
$$\det(C_8^{\mathrm{II}}) = \det(C_8^{\mathrm{III}}) = +1, \quad \det(S_8^{\mathrm{II}}) = \det(S_8^{\mathrm{III}}) = +1, \quad \det(C_8^{\mathrm{IV}}) = \det(S_8^{\mathrm{IV}}) = +1.$$

For $N = 2$ all the DCT and DST matrices are non-eigenorthogonal. For $N > 2$ the DCT and DST matrices except for the DST-I are eigenorthogonal, i.e., their determinants are equal to $+1$. Such matrices are also called the unit matrices [12, 13]. According to Corollary 5.2 of Theorem 5.1 they can be factorized into the product of at most $\frac{N(N-1)}{2}$ elementary Givens–Jacobi rotation matrices, and according to Theorem 5.3 they have *PLUS* factorizations.

5.3 Criteria for evaluation of approximated DCTs/DSTs

In general, the methods for integer approximation of DCT and DST matrices preserve all basic mathematical properties of original real-valued transform matrices such as orthogonality/orthonormality, linearity, symmetry of the basis vectors and recursivity.

In order to evaluate the approximation error between the approximated and original transform matrix, and to measure the difference in performance in data compression, we need some theoretical criteria. For this purpose, the input signal is frequently modeled as a first-order stationary Markov process (Markov-1) with zero-mean, unit variance and adjacent interelement correlation coefficient ρ ranging between zero and one. Then, the input signal $\mathbf{x}$ is defined by a covariance matrix R_x, whose elements are given by

$$[R_x]_{ij} = \rho^{|i-j|}.$$

The matrix R_x is symmetric and Toeplitz. The covariance matrix R_y of the transformed vector $\mathbf{y}$, where $\mathbf{y} = A\mathbf{x}$, is obtained as

$$R_y = A\, R_x\, A^{\mathrm{T}}.$$

Such assumption that the input signal is Markov-1 is generally used to define the theoretical criteria (measures) for the evaluation of approximation error and performance of integer DCTs/DSTs.

5.3.1 Mean-square error

For the evaluation of approximation error between the approximated and original transform matrix we will use *mean-square error* (MSE) defined as follows. Let us assume that U_N is the original transform matrix and $\hat{U}_N$ is its approximation. Then, for a given input vector $\mathbf{x}$ of length N, the error vector is

$$\mathbf{e} = U_N\,\mathbf{x} - \hat{U}_N\,\mathbf{x} = (U_N - \hat{U}_N)\,\mathbf{x} = D\,\mathbf{x}.$$

From above equation, the MSE between the original and approximated transform can be defined as [46]

$$\epsilon = \frac{1}{N} E\left[\mathbf{e}\mathbf{e}^{\mathrm{T}}\right] = \frac{1}{N} E\left[\mathbf{x}^{\mathrm{T}} D^{\mathrm{T}} D\,\mathbf{x}\right] = \frac{1}{N} E\left[\mathrm{Trace}\left\{D\,\mathbf{x}\mathbf{x}^{\mathrm{T}}\,D^{\mathrm{T}}\right\}\right] = \frac{1}{N}\,\mathrm{Trace}\left\{D\,R_x\,D^{\mathrm{T}}\right\}, \tag{5.38}$$

where R_x is the covariance matrix of the input signal $\mathbf{x}$. Trace in (5.38) denotes the trace of a matrix, which is defined as the sum of its diagonal elements. Thus, to maintain the compatibility between the original and approximated transform, the MSE should be minimized.

5.3.2 Transform coding gain

For the transforms used in transform-based coding or data compression applications we need the performance measures to evaluate the coding efficiency of the transform. One of the important measures is *transform coding gain* defined by the covariance matrix R_y as [6, 46]

$$C_g = 10\log_{10}\frac{\frac{1}{N}\sum_{i=0}^{N-1}\sigma_{x_i}^2}{\left(\prod_{i=0}^{N-1}\sigma_{x_i}^2\|f_i\|^2\right)^{\frac{1}{N}}}, \tag{5.39}$$

where N is the number of transform coefficients, $\sigma_{x_i}^2$ is the variance of ith transform coefficient being ith diagonal element of the matrix R_y and $\|f_i\|^2$ is the 2-norm of ith basis function of the transform matrix. Transforms with higher coding gains C_g pack more energy into fewer number of coefficients. As an example, the optimal Karhunen–Loève transform (KLT) has transform coding gain $C_g = 8.8462$, while the DCT-II has transform coding gain $C_g = 8.8259$ for a correlation coefficient $\rho = 0.95$.

5.3.3 Transform efficiency

An alternative measure to the transform coding gain is the *transform efficiency* defined as [28]

$$\eta = \frac{\sum_{i=0}^{N-1}|r_{ii}|}{\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}|r_{ij}|}\,100, \tag{5.40}$$

where r_{ij} are elements of R_y. The transform efficiency measures the decorrelation ability of the transform. The optimal KLT converts signal into completely uncorrelated coefficients and it has transform efficiency $\eta = 100$ for all value of ρ, while the DCT-II has transform efficiency $\eta = 93.9911$ for the correlation coefficient $\rho = 0.95$.

5.4 Methods for integer approximation of DCTs/DSTs

In this section, we present a survey of various methods and design approaches to the integer approximation of DCTs/DSTs. The methods show how to convert real-valued DCTs/DSTs into their integer versions which can be implemented using the simple integer arithmetic

(multiplications and additions or even multiply-free implementation using binary additions and shifts only). The integer transforms are obviously represented by integers with variable bit length. Thus, they can be tuned with different accuracies for a given application. In general, the longer the bit length for representation, the better is the performance. However, this may be at the cost of increased computational complexity. The availability of many integer transforms with different accuracies provides the flexibility to trade off between the performance against low-cost hardware/software implementation and high-speed computation/processing in wireless, hand-held portable devices where CPU capacity, bus width and battery power are limited.

There are two general approaches regarding how to construct the integer transform:

1. To directly replace the real-valued elements of a transform matrix by M-bit integers so that all mathematical properties such as orthogonality/orthonormality and recursive structure of the transform matrix are preserved.
2. To exploit the plane rotation-based sparse matrix factorization of a transform matrix which defines the fast algorithm represented by the corresponding signal flow graph, and to approximate the elements in factored matrices by dyadic rational numbers.

In order to approximate the DCT/DST matrix for a given N, we need both its explicit and numerical form. Almost all methods can be applied to any DCT/DST, although the solutions are not always guaranteed. On the other hand, there are modern methods giving simple and elegant solutions to integer approximation of DCT/DST as long as there exists a plane rotation-based sparse matrix factorization of the transform matrix. As we saw in Section 4.4 such factorizations actually exist for all DCTs/DSTs, and therefore these methods can be regarded as universal.

The approximation methods for generating integer DCTs/DSTs are illustrated in most cases for the 8-point DCT-II since it is the basic processing component in the current international image/video coding standards. The approximation error and performance of the integer transforms are compared to the original real-valued ones based on the standard theoretical criteria defined in Section 5.3 (MSE, transform coding gain and transform efficiency). Higher-order integer DCTs/DSTs can be generated from lower-order ones using the appropriate recursive sparse matrix factorization of the transform matrix. We note that for values of $N > 8$ the complexity of derivation of the integer transform by a given method can increase significantly. Since all mathematical properties of the original real-valued transforms are preserved in the derived integer transforms, for their efficient implementation we can use the corresponding fast algorithm by simply substituting the integer parameters in the signal flow graph.

The description of each method to integer approximation of DCTs/DSTs will include:

- General comments and the main idea of the method with references to bibliography.
- Detailed presentation of the method almost in all cases illustrated for 8-point DCT-II.
- Fast and efficient implementation of forward and inverse integer transform.
- Examples for other DCTs/DSTs if the method can be applied to them.

5.4.1 C-matrix transform

An integer approximation of DCT-II computed via the Walsh–Hadamard transform (WHT), called C-matrix transform (CMT) [19, 22–24], is the first attempt to approximate the real-valued discrete trigonometric transform in the integer domain. At first, Hein and Ahmed [18] showed that DCT-II can be implemented via WHT through a conversion matrix which has sparse-block-diagonal structure. Details of the algorithm for the DCT-II computation via WHT can be found in Section 4.4.3.1. Subsequently, Jones et al. [22] generalized this process and they showed that any even/odd transform (EOT) can be expressed in terms of any other EOT and a conversion matrix. At the same time they obtained an approximation to the DCT-II for $N = 8$ having the following properties:

- It is EOT and orthonormal transform.
- The conversion matrix has only integers as its elements.

From (4.60) it follows that the CMT matrix of order N denoted as C_N^{CMT} is given by

$$\hat{C}_N^{\mathrm{CMT}} = T_N \hat{W}_N, \quad T_N = \hat{C}_N^{\mathrm{II}} \hat{W}_N^{\mathrm{T}}, \tag{5.41}$$

where T_N is the conversion matrix, $\hat{W}_N$ is the sequency ordered WHT matrix and $\hat{C}_N^{\mathrm{II}}$ is the DCT-II matrix, where all matrices, including T_N, have their rows in bit-reversed order. The matrix $\hat{W}_N$ need not be approximated because its elements are ± 1 only. From (5.41) it can be seen that the DCT-II matrix C_N^{II} is approximated indirectly through the conversion matrix T_N whose elements must be integers. The real-valued conversion matrix T_8 both in explicit and numerical form is given respectively as

$$T_8 = \begin{pmatrix}
1 & & & & & & & \\
& 1 & & & & & & \\
& & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & & 0 & & \\
& & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & & & & \\
& & & & \cos\frac{\pi}{8}\cos\frac{\pi}{16} & -\sin\frac{\pi}{8}\sin\frac{\pi}{16} & \sin\frac{\pi}{8}\cos\frac{\pi}{16} & \cos\frac{\pi}{8}\sin\frac{\pi}{16} \\
& & & & \sin\frac{\pi}{8}\sin\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{3\pi}{16} & -\cos\frac{\pi}{8}\sin\frac{3\pi}{16} & \sin\frac{\pi}{8}\cos\frac{3\pi}{16} \\
& & 0 & & -\sin\frac{\pi}{8}\cos\frac{3\pi}{16} & \cos\frac{\pi}{8}\sin\frac{3\pi}{16} & \cos\frac{\pi}{8}\cos\frac{3\pi}{16} & \sin\frac{\pi}{8}\sin\frac{3\pi}{16} \\
& & & & -\cos\frac{\pi}{8}\sin\frac{\pi}{16} & -\sin\frac{\pi}{8}\cos\frac{\pi}{16} & -\sin\frac{\pi}{8}\sin\frac{\pi}{16} & \cos\frac{\pi}{8}\cos\frac{\pi}{16}
\end{pmatrix}$$

$$= \begin{pmatrix}
1 & & & & & & & \\
& 1 & & & & & & \\
& & 0.92388 & 0.38268 & & 0 & & \\
& & -0.38268 & 0.92388 & & & & \\
& & & & 0.90613 & -0.07466 & 0.37533 & 0.18024 \\
& & & & 0.21261 & 0.76818 & -0.51328 & 0.31819 \\
& & 0 & & -0.31819 & 0.51328 & 0.76818 & 0.21261 \\
& & & & -0.18024 & -0.37533 & -0.07466 & 0.90613
\end{pmatrix}.$$

Jones et al. [22] tried to manipulate the matrix T_8 by approximating the ratios in certain rows of the matrix and they derived CMT matrix by trial and error with the following requirements:

- The rows are orthogonal to one another.
- The structure of the conversion matrix should be sparse block diagonal.
- The ratios of elements in a row are close as possible as to those of the conversion matrix.
- The conversion matrix has positive main diagonal.

According to (5.41) the resulting integer versions of conversion matrix denoted by $\bar{T}_8$ and associated CMT matrix C_8^{CMT} are respectively given as

$$\bar{T}_8 = \frac{1}{13}\begin{pmatrix} 13 & & & & & & & \\ & 13 & & & & 0 & & \\ & & 12 & 5 & & & & \\ & & -5 & 12 & & & & \\ & & & & 12 & 0 & 4 & 3 \\ & & & & 0 & 12 & -3 & 4 \\ & 0 & & & -4 & 3 & 12 & 0 \\ & & & & -3 & -4 & 0 & 12 \end{pmatrix},$$

$$C_8^{\text{CMT}} = \frac{1}{\sqrt{8}}\frac{1}{13}\begin{pmatrix} 13 & 13 & 13 & 13 & 13 & 13 & 13 & 13 \\ 19 & 13 & 11 & 5 & -5 & -11 & -13 & -19 \\ 17 & 7 & -7 & -17 & -17 & -7 & 7 & 17 \\ 11 & 5 & -19 & -13 & 13 & 19 & -5 & -11 \\ 13 & -13 & -13 & 13 & 13 & -13 & -13 & 13 \\ 13 & -19 & -5 & 11 & -11 & 5 & 19 & -13 \\ 7 & -17 & 17 & -7 & -7 & 17 & -17 & 7 \\ 5 & -11 & 13 & -19 & 19 & -13 & 11 & -5 \end{pmatrix}.$$

This may be compared to the actual real-valued orthonormal DCT-II matrix C_8^{II} given by

$$C_8^{\text{II}} = \begin{pmatrix} 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 \\ 0.49039 & 0.41573 & 0.27779 & 0.09755 & -0.09755 & -0.27779 & -0.41573 & -0.49039 \\ 0.46194 & 0.19134 & -0.19134 & -0.46194 & -0.46194 & -0.19134 & 0.19134 & 0.46194 \\ 0.41573 & -0.09755 & -0.49039 & -0.27779 & 0.27779 & 0.49039 & 0.09755 & -0.41573 \\ 0.35355 & -0.35355 & -0.35355 & 0.35355 & 0.35355 & -0.35355 & -0.35355 & 0.35355 \\ 0.27779 & -0.49039 & 0.09755 & 0.41573 & -0.41573 & -0.09755 & 0.49039 & -0.27779 \\ 0.19134 & -0.46194 & 0.46194 & -0.19134 & -0.19134 & 0.46194 & -0.46194 & 0.19134 \\ 0.09755 & -0.27779 & 0.41573 & -0.49039 & 0.49039 & -0.41573 & 0.27779 & -0.09755 \end{pmatrix}.$$

MSE approximation error and performance measures of the originally proposed integer CMT for $N = 8$ [22] are shown in Table 5.1. Following the same approach, the integer conversion matrices were derived for $N = 16$ [23] and $N = 32$ [24].

Inspired by the method of integer approximation of DCT-II [28, 36] we describe a more rigorous method to generate new CMTs for $N = 8$ using 6-bit representation ($M = 64$). To construct integer conversion matrix $\bar{T}_8$ we need both the explicit and numerical form of the T_8. Observing the absolute values (magnitudes) of nonzero elements of the conversion matrix T_8 in explicit form we find that

$$
\begin{aligned}
&|t_{00}| = |t_{11}| = 1,\\
&|t_{22}| = |t_{33}| = \cos\frac{\pi}{8}, \quad |t_{23}| = |t_{32}| = \sin\frac{\pi}{8},\\
&|t_{44}| = |t_{77}| = \cos\frac{\pi}{8}\cos\frac{\pi}{16}, \quad |t_{45}| = |t_{76}| = \sin\frac{\pi}{8}\sin\frac{\pi}{16},\\
&|t_{54}| = |t_{67}| = \sin\frac{\pi}{8}\sin\frac{3\pi}{16}, \quad |t_{55}| = |t_{66}| = \cos\frac{\pi}{8}\cos\frac{3\pi}{16},\\
&|t_{46}| = |t_{75}| = \sin\frac{\pi}{8}\cos\frac{\pi}{16}, \quad |t_{47}| = |t_{74}| = \cos\frac{\pi}{8}\sin\frac{\pi}{16},\\
&|t_{56}| = |t_{65}| = \cos\frac{\pi}{8}\sin\frac{3\pi}{16}, \quad |t_{57}| = |t_{64}| = \sin\frac{\pi}{8}\cos\frac{3\pi}{16}.
\end{aligned}
$$

It can be seen that the conversion matrix T_8 is represented by 11 different values. It means that elements with the same magnitudes we can represent by a single variable. Substituting a variable for each nonzero element of T_8 results in a set of 11 variables denoted by $\{a, b, c, d, e, f, g, h, i, j, k\}$. Preserving the signs for elements of T_8 we will search for integer conversion matrix $\bar{T}_8$ in the form:

$$
\bar{T}_8 = \frac{1}{a}\begin{pmatrix}
a & & & & & & & \\
 & a & & & & 0 & & \\
 & & b & c & & & & \\
 & & -c & b & & & & \\
 & & & & d & -e & h & i \\
 & & & & f & g & -j & k \\
 & 0 & & & -k & j & g & f \\
 & & & & -i & -h & -e & d
\end{pmatrix}, \tag{5.42}
$$

where the block matrices $\bar{U}_2$ and $\bar{U}_4$ are given by

$$
\bar{U}_2 = \begin{pmatrix} b & c \\ -c & b \end{pmatrix}, \quad \bar{U}_4 = \begin{pmatrix} d & -e & h & i \\ f & g & -j & k \\ -k & j & g & f \\ -i & -h & -e & d \end{pmatrix}, \tag{5.43}
$$

and variables $a, b, c, d, e, f, g, h, i, j, k \in N$ are assumed to be integers.

Since the real-valued conversion matrix T_8 is orthonormal, $T_8\, T_8^{\mathrm{T}} = I_8$. The same relation must hold for the integer conversion matrix $\bar{T}_8$, i.e., $\bar{T}_8\, \bar{T}_8^{\mathrm{T}} = I_8$. The equation requires that the elements of $\bar{T}_8$ satisfy the following set of algebraic equations

$$df - eg - jh + ik = 0, \tag{5.44}$$

$$dk + ej - hg - if = 0, \tag{5.45}$$

$$b^2 + c^2 = d^2 + e^2 + h^2 + i^2 = f^2 + g^2 + j^2 + k^2 = a^2, \tag{5.46}$$

which must be solved in the integer domain. Equations (5.44) and (5.45) are conditions of orthogonality and they ensure that rows or basis vectors of $\bar{T}_8$ are orthogonal to each other. Equation (5.46) is the normality condition, and it means that the 2-norm of each basis vector of $\bar{T}_8$ is constant and is unity. Finally, in order to make the basis vectors of $\bar{T}_8$ resemble those of real-valued conversion matrix T_8 (magnitudes of integer elements of $\bar{T}_8$ are approximately proportional to those of floating-point elements of T_8), we need to set up the constraints on variables $a, b, c, d, e, f, g, h, i, j, k$. Comparing the magnitudes of elements of the matrix T_8 in numerical form and $\bar{T}_8$ we obtain the following inequalities:

$$d > g > j > h > k > f > i > e > 0, \tag{5.47}$$

$$j > c > h, \tag{5.48}$$

$$b > d, \tag{5.49}$$

$$a > b. \tag{5.50}$$

All integer solutions satisfying (5.44)–(5.46) under constraints given by (5.47)–(5.50) will guarantee that the approximated conversion matrix $\bar{T}_8$ is orthonormal and close to the original conversion matrix T_8. For any solution to integer approximation of CMT for $N = 8$ we will use the notation $\mathrm{CMT}_8\,(a, b, c, d, e, f, g, h, i, j, k)$. If we carry out computer search of the set $\{a, b, c, d, e, f, g, h, i, j, k\}$ satisfying (5.44)–(5.46) under constraints (5.47)–(5.50) using 6-bit representation we find that there exist no solutions. However, by careful analysis and modifying certain constraints we will able to find unique solutions. First, modifying inequality (5.49) to be

$$b \geq d - 1, \tag{5.51}$$

we get the first solution

a	b	c	d	e	f	g	h	i	j	k	a^2
34	30	16	31	1	7	25	13	5	19	11	1156

denoted by $\mathrm{CMT}_8\,(34, 30, 16, 31, 1, 7, 25, 13, 5, 19, 11)$.

Alternatively, modifying inequality in (5.48) to be

$$j > c \geq h - 1, \tag{5.52}$$

Table 5.1. Comparison of the MSE approximation error and performance measures of integer $\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ with the 8-point DCT-II.

$\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$	MSE	C_g	η
8-point DCT-II	–	8.82591	93.99119
$\text{CMT}_8\,(13,12,5,12,0,0,12,4,3,3,4)$	3.300279e−003	8.69805	91.85097
$\text{CMT}_8\,(34,30,16,31,1,7,25,13,5,19,11)$	5.278476e−004	8.77386	92.80060
$\text{CMT}_8\,(39,36,15,35,2,8,30,16,6,19,14)$	2.268015e−004	8.80696	93.18319

we get the second solution

a	b	c	d	e	f	g	h	i	j	k	a^2
39	36	15	35	2	8	30	16	6	19	14	1521

denoted by $\text{CMT}_8\,(39,36,15,35,2,8,30,16,6,19,14)$.

Comparison of the MSE approximation error and performance measures of the new CMTs with the 8-point DCT-II are summarized in Table 5.1. From Table 5.1 we see that the new CMTs are closer to those of the 8-point DCT-II both in terms of MSE error and performance, and they are superior compared to $\text{CMT}_8\,(13,12,5,12,0,0,12,4,3,3,4)$. One can notice the improved performance measures in going from $\text{CMT}_8\,(34,30,16,31,1,7,25,13,5,19,11)$ to $\text{CMT}_8\,(39,36,15,35,2,8,30,16,6,19,14)$.

5.4.1.1 The fast $\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$

The orthonormality property and structure of the real-valued conversion matrix T_8 are preserved in the integer conversion matrix $\bar{T}_8$. This fact allows the use of any fast algorithm for the DCT-II computation via WHT in efficient implementing the CMT. Consequently, for the efficient implementation of $\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ we adopt the signal flow graph from Fig. 4.12. The generalized signal flow graph for the fast $\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ implementation is shown in Fig. 5.6. Although the real-valued block matrix U_4 can be readily factorized into sparse matrices, the problem remains for the existence of sparse matrix factorization of block matrix $\bar{U}_4$ in the integer domain. Therefore, $\bar{U}_4$ is implemented in the form of matrix–vector multiplication.

The fast $\text{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ implementation shown in Fig. 5.6 can be further improved by replacing all integer multiplications with additions and shifts resulting in a multiply-free implementation [24]. Consider the new $\text{CMT}_8\,(39,36,15,35,2,8,30,16,6,19,14)$ with the best performance. Here, each multiplication constant can be expressed in terms of additions and shifts as follows:

$$
\begin{array}{ll}
39 = 32 + 8 - 1 & \text{2 additions and 2 shifts,} \\
36 = 32 + 4 & \text{1 addition and 2 shifts,} \\
15 = 16 - 1 & \text{1 addition and 1 shift,} \\
35 = 32 + 2 + 1 & \text{2 additions and 2 shifts,} \\
2 & \text{1 shift,} \\
8 & \text{1 shift,}
\end{array}
$$

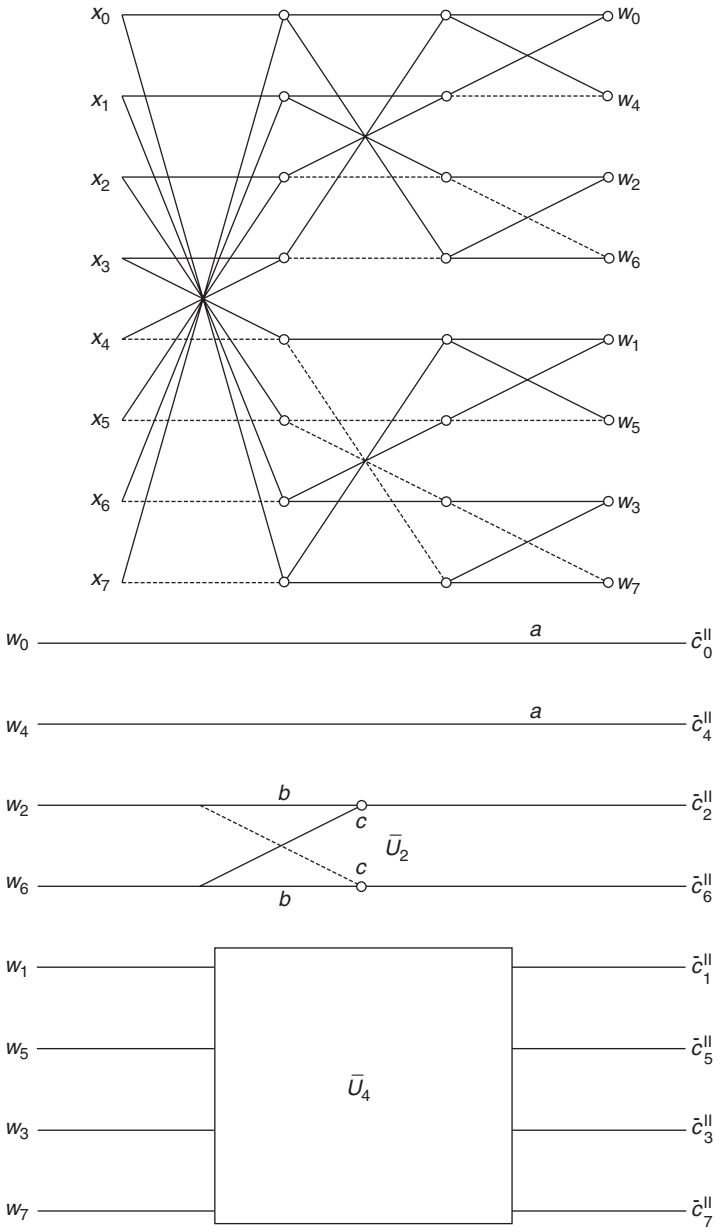


Fig. 5.6. The fast CMT_8 $(a, b, c, d, e, f, g, h, i, j, k)$ implementation.

$30 = 32 - 2$	1 addition and 2 shifts,
16	1 shift,
$6 = 4 + 2$	1 addition and 2 shifts,
$19 = 16 + 2 + 1$	2 additions and 2 shifts,
$14 = 16 - 2$	1 addition and 2 shifts.

Table 5.2. The computational complexity of fast $\mathrm{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$.

$\mathrm{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$	Mults/adds	Adds/shifts
$\mathrm{CMT}_8\,(13,12,5,12,0,0,12,4,3,3,4)$	18/34	50/26
$\mathrm{CMT}_8\,(34,30,16,31,1,7,25,13,5,19,11)$	20/38	64/32
$\mathrm{CMT}_8\,(39,36,15,35,2,8,30,16,6,19,14)$	22/38	54/30

The computational complexity of fast $\mathrm{CMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ in terms of the number of integer multiplications/additions and multiply-free implementation in the form of additions/shifts with respect to the signal flow graph in Fig. 5.6 is shown in Table 5.2.

Note 1: From the relation between DCT-II and DST-II matrices given by (4.11) the fast $\mathrm{SMT}_8\,(a,b,c,d,e,f,g,h,i,j,k)$ for the DST-II computed via WHT can be easily obtained.

Note 2: From the set of DCTs/DSTs (see Chapter 4), the symmetric cosine transform (SCT) and symmetric sine transform (SST) defined by (4.9) and (4.10), respectively, are EOTs and therefore they can be approximated by the same method. However, the conversion matrix T_N for SCT and SST in contrast to the DCT-II will have the following structure [19]

$$T_N = \begin{pmatrix} U_{\frac{N}{2}}^{(1)} & 0 \\ 0 & U_{\frac{N}{2}}^{(2)} \end{pmatrix},$$

i.e., the block matrices $U_{\frac{N}{2}}^{(1)}$ and $U_{\frac{N}{2}}^{(2)}$ are of the same order. Note that the SCT and SST do not have a recursive structure. Consequently, the conversion matrix T_N cannot be generated recursively.

5.4.2 Integer cosine/sine transforms

In this section, we will discuss the method to integer approximation of the DCT-II matrix originally reported in Refs. [25–37]. The main idea of the method was partially utilized to construct the new CMTs in the previous section. In essence, the method is to directly replace the real-valued elements of DCT-II matrix by integers as opposed to the CMT case where the DCT-II matrix is approximated indirectly through the conversion matrix. In general, the method can be applied to any DCT and DST [31, 33] although the integer solutions may not always exist. The resulting integer transforms are referred to as integer cosine/sine transforms of order N (ICT_N/IST_N).

Let A_N be the real-valued DCT/DST matrix with elements $\{a_{ij}\}$, $i,j = 0,1,\ldots,N-1$, and $\bar{A}_N$ its integer approximation with elements $\{\bar{a}_{ij}\}$. The integer approximated ICT_N/IST_N

should satisfy the following additional properties [28, 33, 36, 37]:

- *The orthogonality property*: Rows (columns) of the integer transform matrix are orthogonal to each other.
- *The integer property*: Elements of the approximated transform matrix are integers. In fact, the elements are approximated by rational numbers, i.e., ratios of integers. The unitary nature of these matrices prevents the elements to be just integers.
- *The relationship with real-valued transform matrix*:
  (a) if $|a_{ij}| \geq |a_{ik}|$, then $|\bar{a}_{ij}| \geq |\bar{a}_{ik}|$, $\quad i,j,k = 0,1,\ldots,N-1$;
  (b) $\text{sign}(a_{ij}) = \text{sign}(\bar{a}_{ij})$, $\quad i,j = 0,1,\ldots,N-1$.

The orthogonality property ensures that the forward and inverse $\text{ICT}_N/\text{IST}_N$ have the same transform structure. The integer property eliminates the need for many floating-point arithmetic operations in computing the transform, since the elements of integer transform matrix can be represented by finite number of bits. The orthogonal integer transform matrix it can further be made orthonormal by multiplying it by an appropriate diagonal matrix. The approximation method guarantees that all mathematical properties and the recursive structure (if exists) are preserved in the integer approximated transform matrix. This fact allows any existing fast algorithm to efficiently implement $\text{ICT}_N/\text{IST}_N$. Moreover, the energy packing ability and performance will be close to the original transform. Thus, $\text{ICTs}_N/\text{ISTs}_N$ are functionally compatible to corresponding real-valued DCTs/DSTs.

The method at first is illustrated in detail on the approximation of DCT-II matrix for $N = 8$ together with its possible improvements and simplifications. Since in the recursive sparse matrix factorization (4.50), the DCT-II matrix C_N^{II} requires DCT-II and DCT-IV matrices of half size, the integer approximation of DCT-IV is also presented. Finally, the construction of integer transforms for other DCTs and DSTs is discussed.

5.4.2.1 Integer approximation of DCT-II matrix C_8^{II}

In the following we describe all the steps to convert the real-valued DCT-II matrix C_8^{II} to an integer matrix. Let $C_N^{\text{ICT-II}}$ be the integer approximation of C_8^{II} and ICT_8-II be resulting integer 8-point DCT-II. We will search for the approximated matrix $C_8^{\text{ICT-II}}$ in the form:

$$C_8^{\text{ICT-II}} = Q_8\, V_8, \tag{5.53}$$

where Q_8 is a diagonal matrix with normalization factors on its main diagonal, and V_8 is an integer matrix (V_8 is frequently called the prototype matrix). The procedure to construct the approximated matrix $C_8^{\text{ICT-II}}$ from the corresponding DCT-II matrix C_8^{II} includes the

following steps:

1. Generate the DCT-II matrix C_8^{II} in explicit and numerical form. The matrix C_8^{II} in explicit and numerical form is given respectively as

$$C_8^{\mathrm{II}} = \frac{1}{2}\begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{3\pi}{16} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{\pi}{16} & -\cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & -\sin\frac{\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.35355 \\ 0.49039 & 0.41573 & 0.27779 & 0.09755 & -0.09755 & -0.27779 & -0.41573 & -0.49039 \\ 0.46194 & 0.19134 & -0.19134 & -0.46194 & -0.46194 & -0.19134 & 0.19134 & 0.46194 \\ 0.41573 & -0.09755 & -0.49039 & -0.27779 & 0.27779 & 0.49039 & 0.09755 & -0.41573 \\ 0.35355 & -0.35355 & -0.35355 & 0.35355 & 0.35355 & -0.35355 & -0.35355 & 0.35355 \\ 0.27779 & -0.49039 & 0.09755 & 0.41573 & -0.41573 & -0.09755 & 0.49039 & -0.27779 \\ 0.19134 & -0.46194 & 0.46194 & -0.19134 & -0.19134 & 0.46194 & -0.46194 & 0.19134 \\ 0.09755 & -0.27779 & 0.41573 & -0.49039 & 0.49039 & -0.41573 & 0.27779 & -0.09755 \end{pmatrix}.$$

2. Express the DCT-II matrix C_8^{II} in the form of a matrix of variables, whose elements are assumed to be integers. Observing the absolute values (magnitudes) of the elements of C_8^{II} in explicit form we find that

$$\begin{aligned} &|c_{00}| = |c_{01}| = |c_{02}| = |c_{03}| = |c_{04}| = |c_{05}| = |c_{06}| = |c_{07}| \\ &= |c_{40}| = |c_{41}| = |c_{42}| = |c_{43}| = |c_{44}| = |c_{45}| = |c_{46}| = |c_{47}| = \frac{1}{\sqrt{2}}, \\ &|c_{10}| = |c_{17}| = |c_{32}| = |c_{35}| = |c_{51}| = |c_{56}| = |c_{73}| = |c_{74}| = \cos\frac{\pi}{16}, \\ &|c_{11}| = |c_{16}| = |c_{30}| = |c_{37}| = |c_{53}| = |c_{54}| = |c_{72}| = |c_{75}| = \cos\frac{3\pi}{16}, \\ &|c_{12}| = |c_{15}| = |c_{33}| = |c_{34}| = |c_{50}| = |c_{57}| = |c_{71}| = |c_{76}| = \sin\frac{3\pi}{16}, \\ &|c_{13}| = |c_{14}| = |c_{31}| = |c_{36}| = |c_{52}| = |c_{55}| = |c_{70}| = |c_{77}| = \sin\frac{\pi}{16}, \\ &|c_{20}| = |c_{23}| = |c_{24}| = |c_{27}| = |c_{61}| = |c_{62}| = |c_{65}| = |c_{66}| = \cos\frac{\pi}{8}, \\ &|c_{21}| = |c_{22}| = |c_{25}| = |c_{26}| = |c_{60}| = |c_{63}| = |c_{64}| = |c_{67}| = \sin\frac{\pi}{8}. \end{aligned}$$

We can easily see that the matrix C_8^{II} is represented by seven different values. Let us represent the elements with the same magnitude by a single variable from a set $\{a, b, c, d, e, f, g\}$, where $a, b, c, d, e, f, g \in N$ are integers. This integer property ensures that the matrix V_8 in (5.53) can be represented by finite number of bits. If we substitute the variable for each element of C_8^{II} preserving the signs of the elements, then according to (5.53) the approximated matrix $C_8^{\text{ICT-II}}$ can be expressed as

$$C_8^{\text{ICT-II}} = Q_8 \begin{pmatrix} g & g & g & g & g & g & g & g \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ g & -g & -g & g & g & -g & -g & g \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{pmatrix}. \tag{5.54}$$

3. Find orthogonality conditions under which the ith and jth basis vectors (rows) $\mathbf{v}_i$ and $\mathbf{v}_j$ of the matrix V_8 are orthogonal. Basis vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ are orthogonal if their scalar product satisfies the following relation:

$$\mathbf{v}_i \, \mathbf{v}_j^{\text{T}} = \sum_{k=0}^{7} v_{ik} \, v_{kj} = 0, \quad \forall i \neq j, \quad i,j = 0, 1, \ldots, 7.$$

The orthogonality condition ensures that the integer matrix V_8 is orthogonal. Multiplying it by the diagonal matrix Q_8 ensures that the 2-norm of each basis vector $\mathbf{v}_i$ of V_8 is equal to unity, i.e.,

$$\|\mathbf{v}_i\|_2 = \sqrt{\sum_{k=0}^{7} v_{ik}^2} = 1, \quad i,k = 0, 1, \ldots, 7, \tag{5.55}$$

and it ensures that $C_8^{\text{ICT-II}}$ is orthonormal. From (5.55) it follows that the normalization factors (in general, diagonal elements of Q_8 are irrational numbers) are given by $\frac{1}{\sqrt{q_{ii}}}$, where

$$q_{ii} = \|\mathbf{v}_i\|_2^2. \tag{5.56}$$

The orthogonality condition is equivalent to the evaluation of matrix product

$$V_8 \, V_8^{\text{T}} = [Q_8^{-1}]^2 I_8, \tag{5.57}$$

leading to a set of algebraic equations which are to be solved in the integer domain. The equations are

$$a\,(b-c)-d\,(b+c)=0, \tag{5.58}$$

and

$$\begin{aligned} q_{00} &= q_{44} = 8g^2, \\ q_{11} &= q_{33} = q_{55} = q_{77} = 2\,(a^2+b^2+c^2+d^2), \\ q_{22} &= q_{66} = 4\,(e^2+f^2). \end{aligned} \tag{5.59}$$

Equation (5.58) depends on four variables a, b, c and d. This means that variables e, f and g may be considered independently.

4. Set up the constraints on variables a, b, c, d, e, f, g in the form of inequalities. These ensure that the magnitudes of basis vectors of integer matrix V_8 are roughly proportional to those of the matrix C_8^{II}. With respect to (5.58) and (5.59) we obtain three inequalities as

$$a > b > c > d > 0, \tag{5.60}$$

$$a > e > f > 0, \tag{5.61}$$

$$a > g > 0. \tag{5.62}$$

Since a is the maximal integer number, the variables e,f and g in the last two constraints should be upper bounded by value of a.

5. Find a set of integers $\{a, b, c, d, e, f, g\}$ that satisfy algebraic equation (5.58) under constraints (5.60)–(5.62). To find integer solutions for the set $\{a, b, c, d, e, f, g\}$, a computer search is performed using M-bit representations. In general, the set of algebraic equations may have infinite number of solutions. This implies that infinite number of integer approximated transforms may be generated. The integer transforms defined on the set $\{a, b, c, d, e, f, g\}$ will be denoted as ICT_8-II (a, b, c, d, e, f, g). The resulting solution on the set $\{a, b, c, d, e, f, g\}$ is substituted into the matrix V_8 in equation (5.54). Similarly, the constants q_{ii} given by (5.59) are substituted into the matrix Q_8. All solutions satisfying algebraic equation (5.58) under constraints (5.60)–(5.62) will guarantee that the approximated matrix $C_8^{\text{ICT-II}}$ given by (5.54) is orthonormal and close to the matrix C_8^{II}. The implementation efficiency of ICT_8-II (a, b, c, d, e, f, g) depends on the magnitude of variable with highest value in M-bit representation, i.e., it depends on value of variable a.

In the original proposed construction of integer transforms ICT_8-II (a, b, c, d, e, f, g) [28] the variables g and e,f have been considered independently and fixed to values $g = 1$ and $e = 3, f = 1$. In Table 5.3 are summarized several integer transforms ICT_8-II $(a, b, c, d, 3, 1, 1)$ with the best performances using 3-, 4-, 5-, 6-, 7- and 8-bit representations. One can compare the MSE approximation error and performance measures of integer transforms ICT_8-II $(a, b, c, d, 3, 1, 1)$ with the 8-point DCT-II.

Table 5.3. Comparison of the MSE approximation error and performance measures of originally proposed integer transforms $\text{ICT}_8\text{-II}\,(a, b, c, d, 3, 1, 1)$ with the 8-point DCT-II.

$\text{ICT}_8\text{-II}\,(a, b, c, d, 3, 1, 1)$	MSE	C_g	η
8-point DCT-II	–	8.82591	93.99119
$\text{ICT}_8\text{-II}\,(5, 3, 2, 1, 3, 1, 1)$	2.721681e−003	8.65131	91.12119
$\text{ICT}_8\text{-II}\,(7, 4, 3, 1, 3, 1, 1)$	3.006062e−003	8.61464	90.36893
$\text{ICT}_8\text{-II}\,(10, 9, 6, 2, 3, 1, 1)$	2.060647e−004	8.81413	94.09451
$\text{ICT}_8\text{-II}\,(14, 12, 9, 2, 3, 1, 1)$	4.691150e−004	8.78172	93.39701
$\text{ICT}_8\text{-II}\,(12, 10, 6, 3, 3, 1, 1)$	4.154884e−004	8.78296	92.98370
$\text{ICT}_8\text{-II}\,(15, 12, 8, 3, 3, 1, 1)$	2.059246e−004	8.80668	93.56563
$\text{ICT}_8\text{-II}\,(24, 21, 15, 4, 3, 1, 1)$	2.914385e−004	8.80304	93.69778
$\text{ICT}_8\text{-II}\,(25, 21, 14, 5, 3, 1, 1)$	1.302862e−004	8.81437	93.97981
$\text{ICT}_8\text{-II}\,(25, 24, 16, 5, 3, 1, 1)$	4.693343e−004	8.80224	93.54565
$\text{ICT}_8\text{-II}\,(26, 24, 15, 6, 3, 1, 1)$	2.913951e−004	8.80452	93.69983
$\text{ICT}_8\text{-II}\,(45, 39, 26, 9, 3, 1, 1)$	1.380974e−004	8.81589	94.16741
$\text{ICT}_8\text{-II}\,(45, 42, 28, 9, 3, 1, 1)$	3.317750e−004	8.80878	93.78668
$\text{ICT}_8\text{-II}\,(55, 51, 34, 11, 3, 1, 1)$	3.049383e−004	8.81000	93.84130
$\text{ICT}_8\text{-II}\,(55, 48, 32, 11, 3, 1, 1)$	1.458331e−004	8.81586	94.16161
$\text{ICT}_8\text{-II}\,(65, 57, 38, 13, 3, 1, 1)$	1.524257e−004	8.81576	94.15771
$\text{ICT}_8\text{-II}\,(75, 66, 44, 15, 3, 1, 1)$	1.578933e−004	8.81565	94.15491
$\text{ICT}_8\text{-II}\,(85, 75, 50, 17, 3, 1, 1)$	1.624316e−004	8.81554	94.15281
$\text{ICT}_8\text{-II}\,(120, 105, 70, 24, 3, 1, 1)$	1.492797e−004	8.81581	94.15948
$\text{ICT}_8\text{-II}\,(175, 153, 102, 35, 3, 1, 1)$	1.481648e−004	8.81583	94.16015
$\text{ICT}_8\text{-II}\,(185, 162, 108, 37, 3, 1, 1)$	1.503612e−004	8.81580	94.15886
$\text{ICT}_8\text{-II}\,(230, 201, 134, 46, 3, 1, 1)$	1.475946e−004	8.81584	94.16049
$\text{ICT}_8\text{-II}\,(250, 219, 146, 50, 3, 1, 1)$	1.508895e−004	8.81579	94.15856

For the integer transform $\text{ICT}_8\text{-II}\,(10, 9, 6, 2, 3, 1, 1)$ from Table 5.3, for example, the approximated transform matrix $C_8^{\text{ICT-II}}$ according to (5.54) is given by

$$C_8^{\text{ICT-II}} = Q_8 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 10 & 9 & 6 & 2 & -2 & -6 & -9 & -10 \\ 3 & 1 & -1 & -3 & -3 & -1 & 1 & 3 \\ 9 & -2 & -10 & -6 & 6 & 10 & 2 & -9 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 6 & -10 & 2 & 9 & -9 & -2 & 10 & -6 \\ 1 & -3 & 3 & -1 & -1 & 3 & -3 & 1 \\ 2 & -6 & 9 & -10 & 10 & -9 & 6 & -2 \end{pmatrix},$$

where elements q_{ii} of the diagonal matrix Q_8 are given by

$$q_{00} = q_{44} = \frac{1}{\sqrt{8}}, \quad q_{11} = q_{33} = q_{55} = q_{77} = \frac{1}{\sqrt{442}}, \quad q_{22} = q_{66} = \frac{1}{\sqrt{40}}.$$

Table 5.4. Comparison of the MSE approximation error and performance measures of integer transforms $\text{ICT}_8\text{-II}\,(a,b,c,0,e,f,1)$ with the 8-point DCT-II.

$\text{ICT}_8\text{-II}\,(a,b,c,0,e,f,1)$	MSE	C_g	η
8-point DCT-II	–	8.82591	93.99119
$\text{ICT}_8\text{-II}\,(4,2,2,0,2,1,1)$	6.208293e−003	8.34366	88.05940
$\text{ICT}_8\text{-II}\,(4,2,2,0,4,1,1)$	6.634078e−003	8.31022	88.05968
$\text{ICT}_8\text{-II}\,(4,2,2,0,8,1,1)$	7.899882e−003	8.22918	87.38482

5.4.2.2 Improvements/simplifications of the method

The original method just described to construct integer transforms $\text{ICT}_8\text{-II}\,(a,b,c,d,e,f,g)$ can be further modified to improve/simplify the implementation of integer transforms in terms of computational complexity or required computer memory. A simple approach to improve the computational efficiency of integer transforms $\text{ICT}_8\text{-II}\,(a,b,c,d,e,f,g)$ has been proposed in Ref. [36]. The elements were required to be powers of 2 resulting in a multiply-free implementation using only binary addition and shift operations. To obtain correct solutions of the algebraic equation (5.58), the inequality (5.60) must be slightly modified to the form:

$$a > b \geq c > d = 0$$

with variables g and d fixed to values $g = 1$ and $d = 0$. In Table 5.4 are shown some integer transforms $\text{ICT}_8\text{-II}\,(a,b,c,0,e,f,1)$ with solutions being powers of 2 using 3- and 4-bit representations. It can be seen that the computational efficiency is improved at the cost of degraded performance (the approximated matrix contains zero elements).

The approximated matrix $C_8^{\text{ICT-II}}$ given by (5.53) and (5.54) is the product of diagonal matrix Q_8 and integer matrix V_8. According to (5.59) the diagonal elements of Q_8 are related to the 2-norm of basis vectors of the integer matrix V_8. In general, the diagonal elements of Q_8 are irrational numbers. To eliminate floating-point normalization factors we require the values of diagonal elements q_{ii}, $i = 0,\ldots,7$, to be powers of 2. It means that we require the 2-norm of basis vectors given by (5.59) to be [37]

$$\begin{aligned}
q_{00} &= q_{44} = 8g^2 = 2^{p_1},\\
q_{11} &= q_{33} = q_{55} = q_{77} = 2\,(a^2+b^2+c^2+d^2) = 2^{p_2},\\
q_{22} &= q_{66} = 4\,(e^2+f^2) = 2^{p_3},
\end{aligned}$$

where p_1, p_2 and p_3 are positive integers. Although for the variable g a solution always exists (e.g., $g = 1, 2, 4, \ldots$), the solutions for subsets $\{a,b,c,d\}$ and $\{e,f\}$ under constraints (5.60) and (5.61), respectively, do not exist.

On the other hand, if we require the 2-norm of the basis vectors to be constant, then the diagonal matrix Q_8 can be reduced to a single constant q so that the approximated matrix

$C_8^{\text{ICT-II}}$ can be written in the simplified form as [38]

$$C_8^{\text{ICT-II}} = \frac{1}{\sqrt{q}} \begin{pmatrix} g & g & g & g & g & g & g & g \\ a & b & c & d & -d & -c & -b & -a \\ e & f & -f & -e & -e & -f & f & e \\ b & -d & -a & -c & c & a & d & -b \\ g & -g & -g & g & g & -g & -g & g \\ c & -a & d & b & -b & -d & a & -c \\ f & -e & e & -f & -f & e & -e & f \\ d & -c & b & -a & a & -b & c & -d \end{pmatrix}, \tag{5.63}$$

where

$$q = 8g^2 = 2\,(a^2 + b^2 + c^2 + d^2) = 4\,(e^2 + f^2). \tag{5.64}$$

The constraints on variables a, b, c, d, e, f, g have to be modified to the following inequalities:

$$a > b > c > d > 0, \tag{5.65}$$

$$a > e > b, \quad a > c > f > d, \tag{5.66}$$

$$a > b > g > c. \tag{5.67}$$

Thus, only one normalization constant q is necessary to be stored instead of the diagonal matrix Q_8. Such integer transforms are referred to as integer transforms with constant norm of basis vectors (or with constant self-scalar product) and they are denoted by $\frac{1}{\sqrt{q}}$ ICT$_8$-II (a, b, c, d, e, f, g). To generate the integer transforms $\frac{1}{\sqrt{q}}$ ICT$_8$-II (a, b, c, d, e, f, g) it is necessary to find integer solutions for the subset $\{a, b, c, d\}$ satisfying algebraic equation (5.58) under constraint (5.65), then to find integers $\{e, f\}$ under constraint (5.66), and finally integer g under constraint (5.67) so that the 2-norm of basis vectors given by (5.64) is constant. The complete set of six distinct integer transforms $\frac{1}{\sqrt{q}}$ ICT$_8$-II (a, b, c, d, e, f, g) in 5-, 6-, 7- and 8-bit representations together with MSE approximation error and performances are shown in Table 5.5.

5.4.2.3 The fast ICT$_8$-II (a,b,c,d,e,f,g)

The computational efficiency of the integer transforms ICT$_8$-II (a, b, c, d, e, f, g) can be further improved with a fast computational algorithm.

Consider the recursive sparse matrix factorization of matrix C_8^{II} defined by (4.50). Its general form is the EOT factorization defined by (4.17), but applied recursively. Let $\hat{C}_8^{\text{ICT-II}}$ be the approximated matrix with its rows in bit-reversed order. Then, it can be

Table 5.5. Comparison of the MSE approximation error and performance measures of integer transforms $\frac{1}{\sqrt{q}}$ ICT$_8$-II (a, b, c, d, e, f, g) with constant norm of basis vectors with the 8-point DCT-II.

$\frac{1}{\sqrt{q}}$ ICT$_8$-II (a, b, c, d, e, f, g)	MSE	C_g	η
8-point DCT-II	–	8.82591	93.99119
$\frac{1}{\sqrt{2312}}$ ICT$_8$-II $(24, 20, 12, 6, 23, 7, 17)$	5.278476e−004	8.77386	92.80060
$\frac{1}{\sqrt{57800}}$ ICT$_8$-II $(116, 96, 78, 12, 115, 35, 85)$	1.094566e−003	8.71624	92.72111
$\frac{1}{\sqrt{57800}}$ ICT$_8$-II $(116, 96, 78, 12, 113, 41, 85)$	9.049480e−004	8.73176	92.87965
$\frac{1}{\sqrt{57800}}$ ICT$_8$-II $(116, 96, 78, 12, 97, 71, 85)$	2.301285e−003	8.63349	91.39731
$\frac{1}{\sqrt{121032}}$ ICT$_8$-II $(180, 130, 104, 20, 147, 93, 123)$	1.572639e−003	8.67613	91.05764
$\frac{1}{\sqrt{57800}}$ ICT$_8$-II $(120, 100, 60, 30, 113, 41, 85)$	3.382300e−004	8.78938	92.95941

recursively factorized into the product of orthogonal integer matrices as

$$\hat{C}_8^{\text{ICT-II}} = \begin{pmatrix} \hat{C}_4^{\text{ICT-II}} & 0 \\ 0 & \hat{C}_4^{\text{ICT}-\text{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}$$

$$= \begin{pmatrix} \begin{pmatrix} C_2^{\text{ICT-II}} & 0 \\ 0 & C_2^{\text{ICT-IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \hat{C}_4^{\text{ICT-IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.68}$$

where $C_2^{\text{ICT-II}}$, $C_2^{\text{ICT-IV}}$ and $\hat{C}_4^{\text{ICT-IV}}$ are lower-order integer matrices given by

$$C_2^{\text{ICT-II}} = \begin{pmatrix} g & g \\ g & -g \end{pmatrix}, \quad C_2^{\text{ICT-IV}} J_2 = \begin{pmatrix} f & e \\ -e & f \end{pmatrix},$$

$$\hat{C}_4^{\text{ICT-IV}} J_4 = \begin{pmatrix} d & c & b & a \\ b & d & -a & c \\ -c & -a & -d & b \\ -a & b & -c & d \end{pmatrix}. \tag{5.69}$$

The matrix $C_2^{\text{ICT-II}}$ is an integer DCT-II matrix of order 2, whereas $C_2^{\text{ICT-IV}}$ and $\hat{C}_4^{\text{ICT-IV}}$ are integer DCT-IV matrices of orders 2 and 4, respectively. The advantage of recursive sparse matrix factorization (5.68) is manifold:

- Gives clear insight into recursive structure of the matrix $\hat{C}_N^{\text{ICT-II}}$
- Reveals the relationships among recursively factorized lower-order integer approximated matrices $\hat{C}_{\frac{N}{2}}^{\text{ICT-II}}$ and $\hat{C}_{\frac{N}{2}}^{\text{ICT-IV}}$.

- Simplifies the construction of higher-order integer approximated matrix $\hat{C}_{2N}^{\text{ICT-II}}$ by the composition of lower-order matrices $\hat{C}_{N}^{\text{ICT-II}}$ and $\hat{C}_{N}^{\text{ICT-IV}}$.
- Defines the efficient implementation of integer transforms ICT_8-II, the so-called fast ICT_8-II, which includes lower-order fast integer transforms.

Consider the recursive sparse matrix factorization (5.68). The factored lower-order integer matrix $C_2^{\text{ICT-II}}$ defined by element g, and $C_2^{\text{ICT-IV}}$ defined on the subset $\{e, f\}$ are always orthogonal while the integer matrix $\hat{C}_4^{\text{ICT-IV}}$ defined on the subset $\{a, b, c, d\}$ is orthogonal if

$$a(b-c) - d(b+c) = 0, \quad a > b > c > d > 0.$$

It means that the orthogonality condition for integer transforms $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ given by (5.58) actually corresponds to the orthogonality condition for integer transforms $\text{ICT}_4\text{-IV}\,(a, b, c, d)$. Thus, the integer transforms $\text{ICT}_2\text{-II}\,(g)$, $\text{ICT}_2\text{-IV}\,(e, f)$ and $\text{ICT}_4\text{-IV}\,(a, b, c, d)$ can be solved independently, and this is the reason why in the original construction of $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ [28], the integer solutions for subsets $\{a, b, c, d\}$, $\{e, f\}$ and $\{g\}$ were considered independently. On the other hand, having the integer lower-order matrices $C_2^{\text{ICT-II}}$, $C_2^{\text{ICT-IV}}$ and $\hat{C}_4^{\text{ICT-IV}}$, we can generate the higher-order integer matrix $\hat{C}_8^{\text{ICT-II}}$. Therefore, all solutions for the integer transforms $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ in Tables 5.3 and 5.4 define implicitly the solutions for integer transforms $\text{ICT}_2\text{-II}\,(g)$, $\text{ICT}_2\text{-IV}\,(e, f)$ and $\text{ICT}_4\text{-IV}\,(a, b, c, d)$. We note that the integer DCT-IV matrices $C_2^{\text{ICT-IV}}$ and $C_4^{\text{ICT-IV}}$ are symmetric with constant norm for the basis vectors.

The solutions for integer transforms $\frac{1}{\sqrt{q}}\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ shown in Table 5.5 define implicitly also solutions for integer transforms $\text{ICT}_2\text{-II}\,(g)$, $\text{ICT}_2\text{-IV}\,(e, f)$ and $\text{ICT}_4\text{-IV}\,(a, b, c, d)$, however they are constrained by the condition that the 2-norm of their basis vectors is constant, i.e., $q = 4g^2 = 2(e^2 + f^2) = a^2 + b^2 + c^2 + d^2$. The constraints (5.65)–(5.67) can be expressed in the form of one inequality as

$$a > e > b > g > c > f > d > 0, \tag{5.70}$$

and thus, the known solutions for integer transforms $\text{ICT}_4\text{-IV}\,(a, b, c, d)$ determine the constraints on solutions for integer transforms $\text{ICT}_2\text{-IV}\,(e, f)$ and $\text{ICT}_2\text{-II}\,(g)$. The constraint (5.70) expresses the relations among integer matrices $C_2^{\text{ICT-II}}$, $C_2^{\text{ICT-IV}}$ and $C_4^{\text{ICT-IV}}$, if a single normalization constant q is required.

The recursive sparse matrix factorization (5.68) defines the fast algorithm for the efficient implementation of integer transforms ICT_8-II, called the fast $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$. In order to complete the fast $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ we need to derive a sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$. With reference to the generalized signal flow graph for the fast 8-point DCT-II computation shown in Fig. 4.8 its lower half defines the sparse matrix factorization of DCT-IV matrix $\hat{C}_4^{\text{IV}} J_4$ given by (note that $\cos\frac{\pi}{16} = \sin\frac{7\pi}{16}$

and $\sin\frac{\pi}{16} = \cos\frac{7\pi}{16}$)

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & \cos\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \tag{5.71}$$

Parametrizing the sparse matrix factorization (5.71) we search for the sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$ in the following general product of integer matrices as

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} u & 0 & 0 & v \\ 0 & z & y & 0 \\ 0 & -y & z & 0 \\ -v & 0 & 0 & u \end{pmatrix} \begin{pmatrix} p & s & 0 & 0 \\ r & -p & 0 & 0 \\ 0 & 0 & -p & r \\ 0 & 0 & s & p \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5.72}$$

where the parameters p, r, s, u, v, y, z are integers or dyadic rationals. Note that the dyadic rational number is expressed in the form $\frac{k}{2^m}$, where k, m are integers and k is an odd. Multiplying the factored matrices on right-hand side of (5.72) we get

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} d & c & b & a \\ b & d & -a & c \\ -c & -a & -d & b \\ -a & b & -c & d \end{pmatrix} = \begin{pmatrix} up & s(v-u) & s(u+v) & vp \\ zr & p(z-y) & -p(z+y) & yr \\ -yr & -p(z+y) & -p(z-y) & zr \\ -vp & s(u+v) & -s(v-u) & up \end{pmatrix}, \tag{5.73}$$

and comparing the corresponding elements in the upper half of both matrices in (5.73) leads to the following set of equations:

$$\begin{aligned} up &= d, \quad s(v-u) = c, \quad zr = b, \quad p(z-y) = d, \\ vp &= a, \quad s(u+v) = b, \quad yr = c, \quad p(z+y) = a, \end{aligned} \tag{5.74}$$

where the values of a, b, c, d are given a priori. The equations in (5.74) are mutually dependent through the parameters p and r. To find a unique solution we need to set up first initial values for p and r, and then to find the remaining solutions. For example, consider

the integer transform ICT_8-II $(5, 3, 2, 1, 3, 1, 1)$ from Table 5.3. We have $a = 5$, $b = 3$, $c = 2$ and $d = 1$. Then equations in (5.74) are written as

$$up = 1, \quad s(v-u) = 2, \quad zr = 3, \quad p(z-y) = 1$$

$$vp = 5, \quad s(u+v) = 3, \quad yr = 2, \quad p(z+y) = 5.$$

Setting up the initial values of $p = 1$ and $r = 1$ we find the unique solution as

$$p = 1, \;\; r = 1, \;\; s = \frac{1}{2}, \;\; u = 1, \;\; v = 5, \;\; z = 3, \;\; y = 2,$$

and the sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$ is given by

$$\hat{C}_4^{\text{IV}} J_4 = \begin{pmatrix} 1 & 2 & 3 & 5 \\ 3 & 1 & -5 & 2 \\ -2 & -5 & -1 & 3 \\ -5 & 3 & -2 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 3 & 2 & 0 \\ 0 & -2 & 3 & 0 \\ -5 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{2} & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & \frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

Example 1: Consider the integer transform ICT_8-II $(10, 9, 6, 2, 3, 1, 1)$ from Table 5.3. For $p = 2$ and $r = 3$, we have

$$s = \frac{3}{2}, \;\; u = 1, \;\; v = 5, \;\; z = 3, \;\; y = 2,$$

and the sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$ is given by

$$\hat{C}_4^{\text{IV}} J_4 = \begin{pmatrix} 2 & 6 & 9 & 10 \\ 9 & 2 & -10 & 6 \\ -6 & -10 & -2 & 9 \\ -10 & 9 & -6 & 2 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & 0 & 0 & 5 \\ 0 & 3 & 2 & 0 \\ 0 & -2 & 3 & 0 \\ -5 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 2 & \frac{3}{2} & 0 & 0 \\ 3 & -2 & 0 & 0 \\ 0 & 0 & -2 & 3 \\ 0 & 0 & \frac{3}{2} & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Example 2: Consider the integer transform $\text{ICT}_8\text{-II}\,(4, 2, 2, 0, 2, 1, 1)$ from Table 5.4. For $p = 2$ and $r = 2$, we have

$$s = 1, \;\; u = 0, \;\; v = 2, \;\; z = 1, \;\; y = 1,$$

and the sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$ is given by

$$\hat{C}_4^{\text{IV}} J_4 = \begin{pmatrix} 0 & 2 & 2 & 4 \\ 2 & 0 & -4 & 2 \\ -2 & -4 & 0 & 2 \\ -4 & 2 & -2 & 0 \end{pmatrix}$$

$$= \begin{pmatrix} 0 & 0 & 0 & 2 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -2 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} 2 & 1 & 0 & 0 \\ 2 & -2 & 0 & 0 \\ 0 & 0 & -2 & 2 \\ 0 & 0 & 1 & 2 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Example 3: Consider the integer transform $\frac{1}{\sqrt{2312}}\,\text{ICT}_8\text{-II}\,(24, 20, 12, 6, 23, 7, 17)$ from Table 5.5. For $p = 3$ and $r = 4$, we have

$$s = 2, \;\; u = 2, \;\; v = 8, \;\; z = 5, \;\; y = 3,$$

and the sparse matrix factorization of integer matrix $C_4^{\text{ICT-IV}} J_4$ is given by

$$\hat{C}_4^{\text{IV}} J_4 = \begin{pmatrix} 6 & 12 & 20 & 24 \\ 20 & 6 & -24 & 12 \\ -12 & -24 & -6 & 20 \\ -24 & 20 & -12 & 6 \end{pmatrix}$$

$$= \begin{pmatrix} 2 & 0 & 0 & 8 \\ 0 & 5 & 3 & 0 \\ 0 & -3 & 5 & 0 \\ -8 & 0 & 0 & 2 \end{pmatrix} \begin{pmatrix} 3 & 2 & 0 & 0 \\ 4 & -3 & 0 & 0 \\ 0 & 0 & -3 & 4 \\ 0 & 0 & 2 & 3 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

The generalized signal flow graph for the efficient implementation of integer transforms $\text{ICT}_8\text{-II}$, the fast $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$, is shown in Fig. 5.7. The computational complexity of selected integer transforms $\text{ICT}_8\text{-II}\,(a, b, c, d, e, f, g)$ with the best performance in terms of both the number of integer multiplications/additions and multiply-free implementation with respect to signal flow graph in Fig. 5.7 is summarized in Table 5.6.

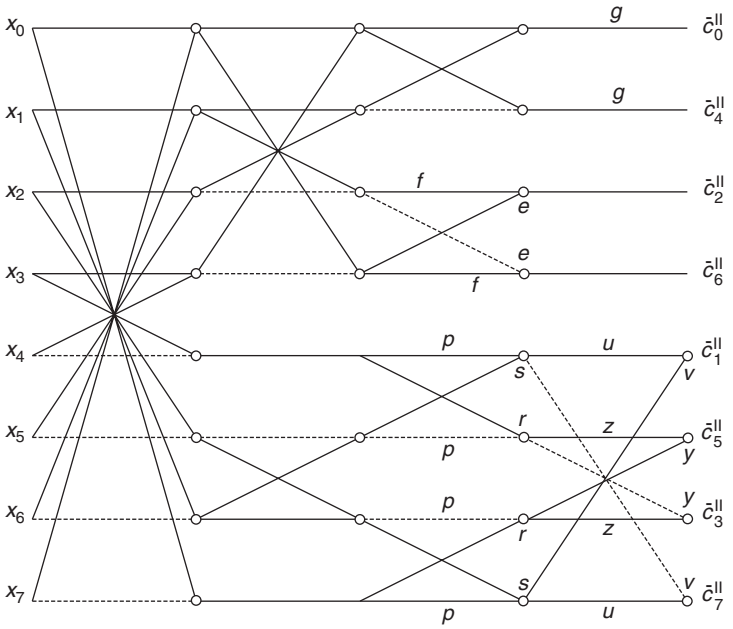


Fig. 5.7. The fast ICT$_8$-II (a, b, c, d, e, f, g) implementation.

Table 5.6. The computational complexity of fast ICT$_8$-II (a, b, c, d, e, f, g).

ICT$_8$-II (a, b, c, d, e, f, g)	Mults/adds	Adds/shifts
ICT$_8$-II $(5, 3, 2, 1, 3, 1, 1)$	10/26	32/10
ICT$_8$-II $(10, 9, 6, 2, 3, 1, 1)$	16/26	36/18
ICT$_8$-II $(4, 2, 2, 0, 2, 1, 1)$	–	24/10
$\frac{1}{\sqrt{2312}}$ ICT$_8$-II $(24, 20, 12, 6, 23, 7, 17)$	22/26	42/24

Note: Based on the relation between DCT-II and DST-II matrices given by (4.11), the fast integer transforms IST$_8$-II (a, b, c, d, e, f, g) can be easily obtained from the corresponding fast integer ICT$_8$-II (a, b, c, d, e, f, g).

5.4.2.4 Integer approximation of DCT-IV matrix C_8^{IV}

Let $C_8^{\text{ICT-IV}}$ be the integer approximation of the DCT-IV matrix C_8^{IV}. Since C_8^{IV} is symmetric with constant norm we will search for the integer approximated matrix $C_8^{\text{ICT-IV}}$ in the form:

$$C_8^{\text{ICT-IV}} = \frac{1}{\sqrt{q}} V_8, \tag{5.75}$$

where V_8 is an integer matrix and q is the normalization constant. The DCT-IV matrix C_8^{IV} in explicit and numerical form is given respectively as

$$C_8^{\mathrm{IV}} = \frac{1}{2}\begin{pmatrix} \cos\frac{\pi}{32} & \cos\frac{3\pi}{32} & \cos\frac{5\pi}{32} & \cos\frac{7\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{5\pi}{32} & \sin\frac{3\pi}{32} & \sin\frac{\pi}{32} \\ \cos\frac{3\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{\pi}{32} & -\sin\frac{5\pi}{32} & -\cos\frac{5\pi}{32} & -\cos\frac{\pi}{32} & -\cos\frac{7\pi}{32} & -\sin\frac{3\pi}{32} \\ \cos\frac{5\pi}{32} & \sin\frac{\pi}{32} & -\cos\frac{7\pi}{32} & -\cos\frac{3\pi}{32} & -\sin\frac{3\pi}{32} & \sin\frac{7\pi}{32} & \cos\frac{\pi}{32} & \sin\frac{5\pi}{32} \\ \cos\frac{7\pi}{32} & -\sin\frac{5\pi}{32} & -\cos\frac{3\pi}{32} & \sin\frac{\pi}{32} & \cos\frac{\pi}{32} & \sin\frac{3\pi}{32} & -\cos\frac{5\pi}{32} & -\sin\frac{7\pi}{32} \\ \sin\frac{7\pi}{32} & -\cos\frac{5\pi}{32} & -\sin\frac{3\pi}{32} & \cos\frac{\pi}{32} & -\sin\frac{\pi}{32} & -\cos\frac{3\pi}{32} & \sin\frac{5\pi}{32} & \cos\frac{7\pi}{32} \\ \sin\frac{5\pi}{32} & -\cos\frac{\pi}{32} & \sin\frac{7\pi}{32} & \sin\frac{3\pi}{32} & -\cos\frac{3\pi}{32} & \cos\frac{7\pi}{32} & \sin\frac{\pi}{32} & -\cos\frac{5\pi}{32} \\ \sin\frac{3\pi}{32} & -\cos\frac{7\pi}{32} & \cos\frac{\pi}{32} & -\cos\frac{5\pi}{32} & \sin\frac{5\pi}{32} & \sin\frac{\pi}{32} & -\sin\frac{7\pi}{32} & \cos\frac{3\pi}{32} \\ \sin\frac{\pi}{32} & -\sin\frac{3\pi}{32} & \sin\frac{5\pi}{32} & -\sin\frac{7\pi}{32} & \cos\frac{7\pi}{32} & -\cos\frac{5\pi}{32} & \cos\frac{3\pi}{32} & -\cos\frac{\pi}{32} \end{pmatrix}$$

$$= \begin{pmatrix} 0.49759 & 0.47847 & 0.44096 & 0.38651 & 0.31720 & 0.23570 & 0.14514 & 0.04901 \\ 0.47847 & 0.31720 & 0.04901 & -0.23570 & -0.44096 & -0.49759 & -0.38651 & -0.14514 \\ 0.44096 & 0.04901 & -0.38651 & -0.47847 & -0.14514 & 0.31720 & 0.49759 & 0.23570 \\ 0.38651 & -0.23570 & -0.47847 & 0.04901 & 0.49759 & 0.14514 & -0.44096 & -0.31720 \\ 0.31720 & -0.44096 & -0.14514 & 0.49759 & -0.04901 & -0.47847 & 0.23570 & 0.38651 \\ 0.23570 & -0.49759 & 0.31720 & 0.14514 & -0.47847 & 0.38651 & 0.04901 & -0.44096 \\ 0.14514 & -0.38651 & 0.49759 & -0.44096 & 0.23570 & 0.04901 & -0.31720 & 0.47847 \\ 0.04901 & -0.14514 & 0.23570 & -0.31720 & 0.38651 & -0.44096 & 0.47847 & -0.49759 \end{pmatrix}.$$

Again the elements of C_8^{IV} can be represented by eight different values. Let these be the set $\{h, i, j, k, l, m, n, o\}$, where $h, i, j, k, l, m, n, o \in N$ are integers. If we substitute a variable for each element of C_8^{IV} preserving the signs of the elements, then according to (5.75) the integer approximated matrix $C_8^{\text{ICT-IV}}$ can be expressed as

$$C_8^{\text{ICT-IV}} = \frac{1}{\sqrt{q}}\begin{pmatrix} h & i & j & k & l & m & n & o \\ i & l & o & -m & -j & -h & -k & -n \\ j & o & -k & -i & -n & l & h & m \\ k & -m & -i & o & h & n & -j & -l \\ l & -j & -n & h & -o & -i & m & k \\ m & -h & l & n & -i & k & o & -j \\ n & -k & h & -j & m & o & -l & i \\ o & -n & m & -l & k & -j & i & -h \end{pmatrix}. \tag{5.76}$$

Table 5.7. The complete set of 17 distinct integer transforms $\frac{1}{\sqrt{q}}$ICT$_8$-IV (h, i, j, k, l, m, n, o) using 6- and 7-bit representations.

h	i	j	k	l	m	n	o	q
42	38	37	32	22	19	10	4	6562
62	61	49	47	37	31	21	5	14971
108	107	81	76	70	61	29	1	44913
93	88	76	66	58	42	34	1	32810
117	106	90	82	59	50	42	1	47490
120	108	104	85	69	52	32	2	52598
94	93	73	70	58	51	26	6	34391
82	79	68	62	50	37	31	6	26299
81	76	64	61	41	38	25	7	23953
87	80	70	65	43	40	25	7	27217
121	111	105	89	69	63	15	8	54927
115	111	109	88	82	59	19	8	55801
126	114	111	96	66	57	30	12	59058
81	80	74	67	53	40	23	13	28033
120	114	103	94	68	57	34	14	56066
121	119	107	97	79	68	19	15	61111
84	81	74	72	42	35	33	18	28679

The orthogonality property leads to a set of algebraic equations

$$\begin{aligned} h\,(i-m) + l\,(i-j) - k\,(m+n) + o\,(j-n) &= 0, \\ h\,(j+n) + l\,(m-n) - k\,(i+j) + o\,(i+m) &= 0, \\ h\,(k+l) - i\,(m+j) + n\,(m-j) + o\,(k-l) &= 0, \end{aligned} \tag{5.77}$$

and the normalization constant q related to the 2-norm of basis vectors is given by

$$q = h^2 + i^2 + j^2 + k^2 + l^2 + m^2 + n^2 + o^2. \tag{5.78}$$

Finally, comparing the corresponding elements of integer matrix V_8 and C_8^{IV} in numerical form (it is sufficient to compare one row only) we obtain the constraints on h, i, j, k, l, m, n, o in the form of the inequality:

$$h > i > j > k > l > m > n > o > 0. \tag{5.79}$$

The complete set of 17 distinct solutions for integer $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (h, i, j, k, l, m, n, o) satisfying (5.77) under constraint (5.79) using 6- and 7-bit representations are shown in Table 5.7. In Table 5.8 are shown some 8-bit solutions. For the integer transform

Table 5.8. Some 8-bit solutions for integer transforms $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (h, i, j, k, l, m, n, o).

h	i	j	k	l	m	n	o	q
134	119	118	98	70	69	11	10	65527
166	162	151	126	118	91	23	10	115311
128	124	119	100	88	87	22	12	68782
126	114	111	96	66	57	30	12	59058
166	152	140	123	89	80	36	12	101150
164	158	136	124	100	74	62	12	105196
188	186	146	140	116	102	52	12	137564
240	228	204	177	150	114	70	12	223069
168	152	148	128	88	76	40	16	104992
162	160	148	134	106	80	46	26	112132
210	204	202	169	147	104	46	26	190298
246	230	229	196	146	115	58	32	243202
252	242	225	202	152	121	69	34	252159
228	222	213	188	148	111	60	36	221102

$\frac{1}{\sqrt{6562}}$ ICT$_8$-IV $(42, 38, 37, 32, 22, 19, 10, 4)$ from Table 5.7, for example, the approximated transform matrix $C_8^{\text{ICT-IV}}$ according to (5.76) is given by

$$C_8^{\text{ICT-IV}} = \frac{1}{\sqrt{6562}} \begin{pmatrix} 42 & 38 & 37 & 32 & 22 & 19 & 10 & 4 \\ 38 & 22 & 4 & -19 & -37 & -42 & -32 & -10 \\ 37 & 4 & -32 & -38 & -10 & 22 & 42 & 19 \\ 32 & -19 & -38 & 4 & 42 & 10 & -37 & -22 \\ 22 & -37 & -10 & 42 & -4 & -38 & 19 & 32 \\ 19 & -42 & 22 & 10 & -38 & 32 & 4 & -37 \\ 10 & -32 & 42 & -37 & 19 & 4 & -22 & 38 \\ 4 & -10 & 19 & -22 & 32 & -37 & 38 & -42 \end{pmatrix}.$$

In the recursive sparse matrix factorization (5.68) we discussed the sparse matrix factorization of integer matrix $\hat{C}_4^{\text{ICT-IV}} J_4$ and the corresponding efficient implementation of integer transforms ICT$_4$-IV (a, b, c, d). The fast computational algorithm for the efficient implementation of integer transforms $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (h, i, j, k, l, m, n, o) can be obtained by using the recursive sparse matrix factorization of the DCT-IV matrix C_8^{IV} presented in Section 4.4.4.

Note: Based on the relation between DCT-IV and DST-IV matrices given by (4.12) the fast integer transforms $\frac{1}{\sqrt{q}}$ IST$_8$-IV (h, i, j, k, l, m, n, o) can be easily obtained from the corresponding fast integer $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (h, i, j, k, l, m, n, o).

5.4.2.5 Integer approximation of the DCT-II matrix C_{16}^{II}

In the following we briefly discuss the construction of integer transforms ICT_{16}-II. Let $C_{16}^{\text{ICT-II}} = Q_{16}\, V_{16}$ be the integer approximation of C_{16}^{II}. The construction of integer transforms ICT_{16}-II can be substantially simplified exploiting the recursive sparse matrix factorization of integer approximated matrix $C_{16}^{\text{ICT-II}}$. There is no need to derive $C_{16}^{\text{ICT-II}}$ in the form of a matrix of variables compared to Refs. [30, 35].

Let $\hat{C}_{16}^{\text{ICT-II}}$ be the integer approximated matrix with its rows in bit-reversed order. Then, it can be recursively factorized into the product of orthogonal integer matrices as

$$\hat{C}_{16}^{\text{ICT-II}} = \begin{pmatrix} \hat{C}_8^{\text{ICT-II}} & 0 \\ 0 & \hat{C}_8^{\text{ICT-IV}}\, J_8 \end{pmatrix} \begin{pmatrix} I_8 & J_8 \\ J_8 & -I_8 \end{pmatrix}, \tag{5.80}$$

where

$$\hat{C}_8^{\text{ICT-IV}}\, J_8 = \begin{pmatrix} o & n & m & l & k & j & i & h \\ k & m & -i & -o & h & -n & -j & l \\ m & h & l & -n & -i & -k & o & j \\ i & -l & o & m & -j & h & -k & n \\ -n & -k & -h & -j & -m & o & l & i \\ -j & o & k & -i & n & l & -h & m \\ -l & -j & n & h & o & -i & -m & k \\ -h & i & -j & k & -l & m & -n & o \end{pmatrix}.$$

The approximated matrix $\hat{C}_8^{\text{ICT-II}}$ is again recursively factorized according to (5.68). Thus, the approximated matrix $\hat{C}_{16}^{\text{ICT-II}}$ can be generated by using the lower-order matrices $\hat{C}_8^{\text{ICT-II}}$ and $\hat{C}_8^{\text{ICT-IV}}$ whose integer solutions are shown in Tables 5.3–5.5 and Tables 5.7–5.8, respectively.

We recall that elements of the diagonal matrix Q_{16} are related to the 2-norm of the basis vectors of $\hat{C}_8^{\text{ICT-II}}$ and $\hat{C}_8^{\text{ICT-IV}}$. To reduce Q_{16} to a single normalization constant q, naturally, it raises the question: Does have exist a single normalization constant q such that ICT_{16}-II will be the integer transform with constant basis vectors? In other words, is there a q such that

$$q = 16g^2 = 8(e^2+f^2) = 4(a^2+b^2+c^2+d^2) = 2(h^2+i^2+j^2+k^2+l^2+m^2+n^2+o^2)? \tag{5.81}$$

In order to get the answer we need to perform a very time-consuming procedure: to find the complete 8-bit solutions for integer transforms $\frac{1}{\sqrt{q}}\,\text{ICT}_8\text{-IV}\,(h,i,j,k,l,m,n,o)$ and choose a solution with the normalization constant q identical to a solution for integer transforms $\frac{1}{\sqrt{q}}\,\text{ICT}_8\text{-II}\,(a,b,c,d,e,f,g)$ summarized in Table 5.5. Fortunately, there

Table 5.9. The complete 8-bit solutions for integer transforms ICT_{16}-II with two normalization constants.

$\frac{1}{\sqrt{q_1}}$ ICT_8-II (a,b,c,d,e,f,g)	$\frac{1}{\sqrt{q_2}}$ $ICT_8 - IV\,(h,i,j,k,l,m,n,o)$
$\frac{1}{\sqrt{166464}}$ (144, 120, 72, 36, 138, 42, 102)	$\frac{1}{\sqrt{171598}}$ (150, 138, 133, 116, 82, 69, 37, 16)
$\frac{1}{\sqrt{226576}}$ (168, 140, 84, 42, 161, 49, 119)	$\frac{1}{\sqrt{231106}}$ (169, 159, 156, 137, 99, 74, 52, 25)
	$\frac{1}{\sqrt{230622}}$ (170, 167, 141, 138, 94, 79, 62, 26)
$\frac{1}{\sqrt{295936}}$ (192, 160, 96, 48, 184, 56, 136)	$\frac{1}{\sqrt{299404}}$ (196, 186, 165, 144, 124, 83, 80, 8)
$\frac{1}{\sqrt{374544}}$ (216, 180, 108, 54, 207, 63, 153)	$\frac{1}{\sqrt{379788}}$ (228, 209, 190, 164, 128, 104, 63, 8)
	$\frac{1}{\sqrt{394182}}$ (231, 213, 191, 177, 117, 95, 86, 21)
	$\frac{1}{\sqrt{378358}}$ (218, 213, 181, 177, 119, 95, 89, 33)
$\frac{1}{\sqrt{462400}}$ (240, 200, 120, 60, 230, 70, 170)	$\frac{1}{\sqrt{473382}}$ (246, 237, 204, 186, 150, 111, 93, 18)
$\frac{1}{\sqrt{462400}}$ (240, 200, 120, 60, 226, 82, 170)	

exists a more efficient procedure. Consider all unique solutions for integer transforms $\frac{1}{\sqrt{q}}$ ICT_4-IV (a,b,c,d) from Table 5.5 given as:

a	b	c	d
24	20	12	6
116	96	78	12
180	130	104	20

Then, for the given set $\{a,b,c,d\}$ we must find solutions for $\frac{1}{\sqrt{q}}$ ICT_8-IV (h,i,j,k,l,m,n,o) which satisfy equations (5.77) under the constraints:

$$h > a > i > j > b > k > l > c > m > n > d > o > 0, \tag{5.82}$$

and such that the 2-norm of basis vectors is constant, i.e., equation (5.81) holds. The inequality (5.82) actually defines the relation between integer transforms $\frac{1}{\sqrt{q}}$ ICT_4-IV (a,b,c,d) and $\frac{1}{\sqrt{q}}$ ICT_8-IV (h,i,j,k,l,m,n,o) if the single normalization constant q is required. A computer search reveals that the diagonal matrix Q_{16} can only be reduced to two normalization constants q_1 for integer transforms $\frac{1}{\sqrt{q_1}}$ ICT_8-II (a,b,c,d,e,f,g) and q_2 for integer transforms $\frac{1}{\sqrt{q_2}}$ ICT_8-IV (h,i,j,k,l,m,n,o). The complete 8-bit solutions with two normalization constants q_1 and q_2 are shown in Table 5.9. As can be seen from Table 5.9 for a given solution $\{a,b,c,d\}$ of $\frac{1}{\sqrt{q_1}}$ ICT_8-II (a,b,c,d,e,f,g) there may be more than one solution $\{h,i,j,k,l,m,n,o\}$ for $\frac{1}{\sqrt{q_2}}$ ICT_8-IV (h,i,j,k,l,m,n,o) satisfying constraints (5.82) and vice versa.

It is important to note that by constructing the integer transforms ICT_{16}-II we get at the same time the integer transforms ICT_2-II (g), ICT_2-IV (e,f), ICT_4-IV (a,b,c,d), ICT_8-II (a,b,c,d,e,f,g) and ICT_8-IV (h,i,j,k,l,m,n,o).

Finally, the recursive sparse matrix factorization of integer approximated matrix $C_{16}^{\text{ICT-II}}$ given by (5.80) defines the fast ICT_{16}-II. Due to the recursive structure of matrix $C_{16}^{\text{ICT-II}}$, the upper half of generalized signal flow graph representing the fast ICT_{16}-II will correspond to the signal flow graph shown in Fig. 5.7. To complete the lower part of fast ICT_{16}-II we need only derive a sparse matrix factorization of the integer approximated matrix $C_8^{\text{ICT-IV}}$.

5.4.2.6 General procedure to construct integer DCTs/DSTs

The integer approximation method previously described in detail to construct integer transforms ICT_N-II/IST_N-II and ICT_N-IV/IST_N-IV from corresponding real-valued DCT-II/DST-II and DCT-IV/DST-IV, respectively, can be applied to any DCT and DST and in general, to any discrete sinusoidal transform such as the discrete Fourier transform (DFT) and discrete Hartley transform (DHT) [37].

In the following are summarized all steps of the general procedure to construct integer trigonometric transform including possible improvements and simplifications. Let C_N/S_N be the real-valued DCT/DST matrix. Then the general procedure for construction of approximated integer transform is as follows:

1. Based on the definition of DCT/DST generate the transform matrix C_N/S_N both in explicit and numerical forms. The corresponding integer approximated transform matrix $C_N^{\text{ICT}}/S_N^{\text{IST}}$ can be expressed in product form $Q_N\,V_N$, where Q_N is a diagonal matrix whose elements on main diagonal represent normalization factors (generally they are irrational numbers) and V_N is an integer matrix.
2. Represent the elements of transform matrix C_N/S_N in explicit form by a set of variables whose values are assumed to be integers. Elements with the same absolute value are represented by a single variable. Substituting a variable for each element of C_N/S_N and preserving the signs of the elements results in an integer matrix V_N.
3. Find the orthogonality conditions for the variables under which the integer matrix V_N is orthogonal. This is equivalent to the evaluation of matrix product $V_N\,V_N^{\text{T}} = [Q_N^{-1}]^2 I_N$ which leads to a set of algebraic equations. The elements of the diagonal matrix Q_N are related to the 2-norm of the rows of V_N, and they are given by $\frac{1}{\sqrt{q_{ii}}}$, where constants $q_{ii} = \|\mathbf{v}_i\|_2^2$, $\|\mathbf{v}_i\|_2^2 = \sum_{k=0}^{N-1} v_{ik}^2$, $i,j = 0,1,\ldots,N-1$. Multiplying the integer matrix V_N by diagonal matrix Q_N the approximated matrix $C_N^{\text{ICT}}/S_N^{\text{IST}}$ becomes orthonormal.
4. Find the constraints in the form of inequalities on the set of variables representing integer matrix V_N by ordering the magnitudes of elements of transform matrix C_N/S_N.
5. Find the integer solutions for the set of variables satisfying the orthogonality conditions, i.e., satisfying the set of algebraic equations under constraints using M-bit representation. For each integer solution compute the elements of diagonal matrix

Q_N. The method can be further modified to improve and simplify the implementation of integer transforms:

- Find integer solutions being powers of 2 to obtain directly the multiply-free implementation.
- To reduce the diagonal matrix Q_N to a single normalization constant q, find integer solutions such that the 2-norm of basis vectors is constant.

6. Maximize the computational efficiency of the integer transform by a fast computational algorithm. The existence of a (recursive) sparse matrix factorization of the transform matrix C_N/S_N defining the fast algorithm enables the efficient implementation of integer transforms. Moreover, the recursive sparse matrix factorization of approximated transform matrix simplifies the construction of higher-order approximated matrix simply by the composition of lower-order integer approximated matrices.

5.4.2.7 Examples of construction of integer DCTs/DSTs

To conclude this section, we present the construction of integer transforms for the remaining DCTs and DSTs for $N = 8$. Examples show that the solutions for a given approximated transform may not always exist.

Integer approximation of DCT-I matrix C_9^{I}

According to definition (4.1) the transform matrix C_9^{I} in explicit and numerical form is given respectively as

$$C_9^{\mathrm{I}} = \frac{1}{2}\begin{pmatrix}
\frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\
\frac{1}{\sqrt{2}} & \cos\frac{\pi}{8} & \cos\frac{\pi}{4} & \sin\frac{\pi}{8} & 0 & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \cos\frac{\pi}{4} & 0 & -\cos\frac{\pi}{4} & -1 & -\cos\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & \sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & -\cos\frac{\pi}{8} & 0 & \cos\frac{\pi}{8} & \cos\frac{\pi}{4} & -\sin\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & 0 & -1 & 0 & 1 & 0 & -1 & 0 & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & \cos\frac{\pi}{8} & 0 & -\cos\frac{\pi}{8} & \cos\frac{\pi}{4} & \sin\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\cos\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & -1 & \cos\frac{\pi}{4} & 0 & -\cos\frac{\pi}{4} & \frac{1}{\sqrt{2}} \\
\frac{1}{\sqrt{2}} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{4} & -\sin\frac{\pi}{8} & 0 & \sin\frac{\pi}{8} & -\cos\frac{\pi}{4} & \cos\frac{\pi}{8} & -\frac{1}{\sqrt{2}} \\
\frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{2}
\end{pmatrix}$$

$$= \begin{pmatrix}
0.25000 & 0.35355 & 0.35355 & 0.33555 & 0.35355 & 0.35355 & 0.35355 & 0.35355 & 0.25000 \\
0.35355 & 0.46194 & 0.35355 & 0.19134 & 0.00000 & -0.19134 & -0.35355 & -0.46194 & -0.35355 \\
0.35355 & 0.35355 & 0.00000 & -0.35355 & -0.50000 & -0.35355 & 0.00000 & 0.35355 & 0.35355 \\
0.35355 & 0.19134 & -0.35355 & -0.46194 & 0.00000 & 0.46194 & -0.35355 & -0.19134 & -0.35355 \\
0.35355 & 0.00000 & -0.50000 & 0.00000 & 0.50000 & 0.00000 & -0.50000 & 0.00000 & 0.35355 \\
0.35355 & -0.19134 & -0.35355 & 0.46194 & 0.00000 & -0.46194 & -0.35355 & 0.19134 & -0.35355 \\
0.35355 & -0.35355 & 0.00000 & 0.35355 & -0.50000 & 0.35355 & 0.00000 & -0.35355 & 0.35355 \\
0.35355 & -0.46194 & 0.35355 & -0.19134 & 0.00000 & 0.19134 & -0.35355 & 0.46194 & -0.35355 \\
0.25000 & -0.35355 & 0.35355 & -0.35355 & 0.35355 & -0.35355 & 0.35355 & -0.35355 & 0.25000
\end{pmatrix}.$$

The elements of C_9^{I} can be represented by five different values, i.e., by the set $\{a, b, c, d, e\}$, where $a, b, c, d, e \in N$ are integers. Substituting a variable for each element of C_9^{I} and preserving the signs of the elements, the approximated matrix $C_9^{\text{ICT-I}}$ is expressed as

$$C_9^{\text{ICT-I}} = Q_9\, V_9 = Q_9 \begin{pmatrix} d & c & c & c & c & c & c & c & d \\ c & b & c & e & 0 & -e & -c & -b & -c \\ c & c & 0 & -c & -a & -c & 0 & c & c \\ c & e & -c & -b & 0 & b & c & -e & -c \\ c & 0 & -a & 0 & a & 0 & -a & 0 & c \\ c & -e & -c & b & 0 & -b & c & e & -c \\ c & -c & 0 & c & -a & c & 0 & -c & c \\ c & -b & c & -e & 0 & e & -c & b & -c \\ d & -c & c & -c & c & -c & c & -c & d \end{pmatrix},$$

where diagonal elements of the matrix Q_9 are given by

$$\begin{aligned} q_{00} &= q_{88} = 7c^2 + 2d^2, \\ q_{11} &= q_{33} = q_{55} = q_{77} = 2(b^2 + 2c^2 + e^2), \\ q_{22} &= q_{66} = a^2 + 6c^2, \\ q_{44} &= 3a^2 + 2c^2. \end{aligned}$$

The orthogonality condition of $C_9^{\text{ICT-I}}$ leads to a set of algebraic equations

$$\begin{aligned} &c(2d - a) = 0, \quad \text{or} \quad 2d - a = 0, \\ &2c^2 - a^2 = 0, \\ &2d^2 - c^2 = 0, \end{aligned}$$

and the constraint on variables a, b, c, d, e is given by the inequality:

$$a > b > c > d > e > 0.$$

From the set of algebraic equations it follows that $a = \sqrt{2}c$ and $c = \sqrt{2}d$. In number theory it is well known that $\sqrt{2}$ is irrational number and, therefore, a and c cannot be integers. Consequently, the integer solutions for the set $\{a, b, c, d, e\}$ do not exist, i.e., the matrix C_9^{I} cannot be integer approximated by the method.

Integer approximation of DST-I matrix S_7^{I}

According to definition (4.5) the transform matrix S_7^{I} in explicit and numerical form is given respectively as

$$S_7^{\mathrm{I}} = \frac{1}{2}\begin{pmatrix} \sin\frac{\pi}{8} & \sin\frac{\pi}{4} & \cos\frac{\pi}{8} & 1 & \cos\frac{\pi}{8} & \sin\frac{\pi}{4} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{4} & 1 & \sin\frac{\pi}{4} & 0 & -\sin\frac{\pi}{4} & -1 & -\sin\frac{\pi}{4} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{4} & -\sin\frac{\pi}{8} & -1 & -\sin\frac{\pi}{8} & \sin\frac{\pi}{4} & \cos\frac{\pi}{8} \\ 1 & 0 & -1 & 0 & 1 & 0 & -1 \\ \cos\frac{\pi}{8} & -\sin\frac{\pi}{4} & -\sin\frac{\pi}{8} & 1 & -\sin\frac{\pi}{8} & -\sin\frac{\pi}{4} & \cos\frac{\pi}{8} \\ \sin\frac{\pi}{4} & -1 & \sin\frac{\pi}{4} & 0 & -\sin\frac{\pi}{4} & 1 & -\sin\frac{\pi}{4} \\ \sin\frac{\pi}{8} & -\sin\frac{\pi}{4} & \cos\frac{\pi}{8} & -1 & \cos\frac{\pi}{8} & -\sin\frac{\pi}{4} & \sin\frac{\pi}{8} \end{pmatrix}$$

$$= \begin{pmatrix} 0.19134 & 0.35355 & 0.46194 & 0.50000 & 0.46194 & 0.35355 & 0.19134 \\ 0.35355 & 0.50000 & 0.35355 & 0.00000 & -0.35355 & -0.50000 & -0.35355 \\ 0.46194 & 0.35355 & -0191340 & -0.50000 & -0.19134 & 0.35355 & 0.46194 \\ 0.50000 & 0.00000 & -0.50000 & 0.00000 & 0.50000 & 0.00000 & -0.50000 \\ 0.46194 & -0.35355 & -0.19134 & 0.50000 & -0.19134 & -0.35355 & 0.46194 \\ 0.35355 & -0.50000 & 0.35355 & 0.00000 & -0.35355 & 0.50000 & -0.35355 \\ 0.19134 & -0.35355 & 0.46194 & -0.50000 & 0.46194 & -0.35355 & 0.19134 \end{pmatrix}.$$

The elements of S_7^{I} can be represented by four different values, i.e., by the set $\{a, b, c, d\}$, where $a, b, c, d \in N$ are integers. Substituting a variable for each element of S_7^{I} and preserving the signs of the elements, the approximated matrix $S_7^{\mathrm{IST\text{-}I}}$ is expressed as

$$S_7^{\mathrm{IST\text{-}I}} = Q_7\, V_7 = Q_7 \begin{pmatrix} a & b & c & d & c & b & a \\ b & d & b & 0 & -b & -d & -b \\ c & b & -a & -d & -a & b & c \\ d & 0 & -d & 0 & d & 0 & -d \\ c & -b & -a & d & -a & -b & c \\ b & -d & b & 0 & -b & d & -b \\ a & -b & c & -d & c & -b & a \end{pmatrix},$$

where diagonal elements of the matrix Q_7 are given by

$$q_{00} = q_{22} = q_{44} = q_{66} = 2(a^2 + b^2 + c^2) + d^2,$$

$$q_{11} = q_{55} = 4b^2 + 2d^2,$$

$$q_{33} = 4d^2.$$

The orthogonality condition of $S_7^{\text{IST-I}}$ leads to a set of algebraic equations

$$2b^2 - d^2 = 0,$$
$$2a^2 - 2\,b^2 + 2\,c^2 - d^2 = 0 \quad \text{or} \quad 2a^2 - 2b^2 + c^2 = 0,$$

and the constraint on variables a, b, c, d is given by the inequality as

$$d > c > b > a > 0.$$

Again, from the set of algebraic equations it follows that $d = \sqrt{2}b$. Since $\sqrt{2}$ is irrational number, d cannot be integer and, therefore, the matrix S_7^{I} cannot be integer approximated by the method.

Integer approximation of SCT matrix $\tilde{C}_8^{\text{I}}$

According to definition (4.9) the transform matrix $\tilde{C}_8^{\text{I}}$ in explicit and numerical form is given respectively as

$$\tilde{C}_8^{I} = \sqrt{\frac{2}{7}} \begin{pmatrix} \frac{1}{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{2} \\ \frac{1}{\sqrt{2}} & \cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{3\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{2\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{3\pi}{7} & -\cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & -\cos\frac{3\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{\pi}{7} & -\cos\frac{\pi}{7} & \cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\cos\frac{\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{3\pi}{7} & -\cos\frac{3\pi}{7} & \cos\frac{2\pi}{7} & -\cos\frac{\pi}{7} & \frac{1}{\sqrt{2}} \\ \frac{1}{2} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & -\frac{1}{2} \end{pmatrix}$$

$$= \begin{pmatrix} 0.26726 & 0.37796 & 0.37796 & 0.37796 & 0.37796 & 0.37796 & 0.37796 & 0.26726 \\ 0.37796 & 0.48159 & 0.33327 & 0.11894 & -0.11894 & -0.33327 & -0.48159 & -0.37796 \\ 0.37796 & 0.33327 & -0.11894 & -0.48159 & -0.48159 & -0.11894 & 0.33327 & 0.37796 \\ 0.37796 & 0.11894 & -0.48159 & -0.33327 & 0.33327 & 0.48159 & -0.11894 & -0.37796 \\ 0.37796 & -0.11894 & -0.48159 & 0.33327 & 0.33327 & -0.48159 & -0.11894 & 0.37796 \\ 0.37796 & -0.33327 & -0.11894 & 0.48159 & -0.48159 & 0.11894 & 0.33327 & -0.37796 \\ 0.37796 & -0.48159 & 0.33327 & -0.11894 & -0.11894 & 0.33327 & -0.48159 & 0.37796 \\ 0.26726 & -0.37796 & 0.37796 & -0.37796 & 0.37796 & -0.37796 & 0.37796 & -0.26726 \end{pmatrix}.$$

The elements of $\tilde{C}_8^{\text{I}}$ can be represented by five different values, i.e., by the set $\{a, b, c, d, e\}$, where $a, b, c, d, e \in N$ are integers. If in addition we require the norm of basis vectors to be constant, then Q_8 is reduced to a single constant q. Substituting a variable for each

element of $\tilde{C}_8^{\mathrm{I}}$ and preserving the signs of the elements, the approximated matrix C_8^{ISCT} is expressed as

$$C_8^{\mathrm{ISCT}} = \frac{1}{\sqrt{q}} \begin{pmatrix} b & d & d & d & d & d & d & b \\ d & e & c & a & -a & -c & -e & -d \\ d & c & -a & -e & -e & -a & c & d \\ d & a & -e & -c & c & e & -a & -d \\ d & -a & -e & c & c & -e & -a & d \\ d & -c & -a & e & -e & a & c & -d \\ d & -e & c & -a & -a & c & -e & d \\ b & -d & d & -d & d & -d & d & -b \end{pmatrix}.$$

where

$$q = 2(b^2 + 3d^2) = 2(a^2 + c^2 + d^2 + e^2).$$

The orthogonality condition for C_8^{ISCT} is equivalent to a set of algebraic equations

$$(b + c) - (a + e) = 0,$$
$$d^2 + e(a - c) - ac = 0,$$

and the constraint on variables a, b, c, d, e is given by

$$e > d > c > b > a > 0.$$

4- and 5-bit solutions for integer transforms $\frac{1}{\sqrt{q}}$ ISCT_8 (a, b, c, d, e) are shown in Table 5.10. You can note that some integer solutions are multiples of the corresponding unique solutions.

Although the matrix $\tilde{C}_8^{\mathrm{I}}$ does not have a recursive structure it is an EOT. Let $\hat{C}_8^{\mathrm{ISCT}}$ be the approximated matrix with its rows in bit-reversed order. In order to get an efficient implementation of $\frac{1}{\sqrt{q}}$ ISCT_8 (a, b, c, d, e), called the fast ISCT_8 (a, b, c, d, e), according to (4.17), and (4.23) and (4.24), the approximated matrix $\hat{C}_8^{\mathrm{ISCT}}$ can be factorized into the following nonrecursive matrix product:

$$C_8^{\mathrm{ISCT}} = \frac{1}{\sqrt{q}} \begin{pmatrix} b & d & d & d & & & & \\ d & -a & -e & c & & 0 & & \\ d & c & -a & -e & & & & \\ d & -e & c & -a & & & & \\ & & & & a & c & e & d \\ & & & & e & -a & -c & d \\ & & 0 & & -c & -e & a & d \\ & & & & -d & d & -d & b \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}.$$

Table 5.10. 4- and 5-bit solutions for integer transforms $\frac{1}{\sqrt{q}}\mathrm{ISCT}_8\ (a, b, c, d, e)$.

a	b	c	d	e	q
1	6	9	11	14	798
1	8	16	19	23	2294
1	11	13	17	23	1976
2	5	7	8	10	434
2	7	14	16	19	1634
2	9	23	26	30	4218
2	12	18	22	28	3192
2	15	17	22	30	3354
3	6	10	11	13	798
3	8	19	21	24	2774
3	9	11	13	17	1176
4	7	13	14	16	1274
4	9	24	26	29	4218
4	10	14	16	20	1736
4	13	15	18	24	2282
4	15	20	24	31	3906
5	8	16	17	19	1862
5	11	17	19	23	2408
5	12	13	15	20	1638
5	14	18	21	27	3038
5	17	19	23	31	3752
6	9	19	20	22	2562
6	12	20	22	26	3192
6	15	21	24	30	3906
7	10	22	23	25	3374
7	13	23	25	29	4088
8	11	25	26	28	4298
8	15	20	22	27	3354
8	16	17	19	25	2678
9	12	28	29	31	5334
9	17	19	21	27	3224
10	15	17	18	22	2394
10	18	21	23	29	3822
11	19	23	25	31	4472
13	18	22	23	27	3822

Integer approximation of SST matrix $\tilde{S}_8^{\mathrm{I}}$

According to definition (4.10) the transform matrix $\tilde{S}_8^{\mathrm{I}}$ in explicit and numerical form is given below.

$$\tilde{S}_8^{\mathrm{I}} = \frac{\sqrt{2}}{3}\begin{pmatrix} \sin\frac{\pi}{9} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{4\pi}{9} & \sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{9} \\ \sin\frac{2\pi}{9} & \sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & \sin\frac{\pi}{9} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & -\sin\frac{2\pi}{9} \\ \sin\frac{\pi}{3} & \sin\frac{\pi}{3} & 0 & -\sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & \sin\frac{\pi}{3} \\ \sin\frac{4\pi}{9} & \sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{2\pi}{9} & \sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & -\sin\frac{4\pi}{9} \\ \sin\frac{4\pi}{9} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & \sin\frac{2\pi}{9} & -\sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & \sin\frac{4\pi}{9} \\ \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} & 0 & \sin\frac{\pi}{3} & -\sin\frac{\pi}{3} \\ \sin\frac{2\pi}{9} & -\sin\frac{4\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{\pi}{9} & -\sin\frac{\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & \sin\frac{2\pi}{9} \\ \sin\frac{\pi}{9} & -\sin\frac{2\pi}{9} & \sin\frac{\pi}{3} & -\sin\frac{4\pi}{9} & \sin\frac{4\pi}{9} & -\sin\frac{\pi}{3} & \sin\frac{2\pi}{9} & -\sin\frac{\pi}{9} \end{pmatrix}$$

$$= \begin{pmatrix} 0.16123 & 0.30301 & 0.40825 & 0.46424 & 0.46424 & 0.40825 & 0.30301 & 0.16123 \\ 0.30301 & 0.46424 & 0.40825 & 0.16123 & -0.16123 & -0.40825 & -0.46424 & -0.30301 \\ 0.40825 & 0.40825 & 0.00000 & -0.40825 & -0.40825 & -0.00000 & 0.40825 & 0.40825 \\ 0.46424 & 0.16123 & -0.40825 & -0.30301 & 0.30301 & 0.40825 & -0.16123 & -0.46424 \\ 0.46424 & -0.16123 & -0.40825 & 0.30301 & 0.30301 & -0.40825 & -0.16123 & 0.46424 \\ 0.40825 & -0.40825 & -0.00000 & 0.40825 & -0.40825 & -0.00000 & 0.40825 & -0.40825 \\ 0.30301 & -0.46424 & 0.40825 & -0.16123 & -0.16123 & 0.40825 & -0.46424 & 0.30301 \\ 0.16123 & -0.30301 & 0.40825 & -0.46424 & 0.46424 & -0.40825 & 0.30301 & -0.16123 \end{pmatrix}.$$

The elements of $\tilde{S}_8^{\mathrm{I}}$ can be represented by four different values, i.e., by the set $\{a, b, c, d\}$, where $a, b, c, d \in N$ are integers. If we require the norm of basis vectors to be constant, then Q_8 is reduced to a single constant q. Substituting a variable for each element of $\tilde{S}_8^{\mathrm{I}}$ and preserving the signs of the elements, the approximated matrix S_8^{ISST} is expressed as

$$S_8^{\mathrm{ISST}} = \frac{1}{\sqrt{q}}\begin{pmatrix} a & b & c & d & d & c & b & a \\ b & d & c & a & -a & -c & -d & -b \\ c & c & 0 & -c & -c & 0 & c & c \\ d & a & -c & -b & b & c & -a & -d \\ d & -a & -c & b & b & -c & -a & d \\ c & -c & 0 & c & -c & 0 & c & -c \\ b & -d & c & -a & -a & c & -d & b \\ a & -b & c & -d & d & -c & b & -a \end{pmatrix}.$$

where

$$q = 2(a^2 + b^2 + c^2 + d^2) = 6c^2.$$

The orthogonality condition for S_8^{ISST} is equivalent to a set of algebraic equations

$$\begin{aligned} &c(a + b - d) = 0 \quad \text{or} \quad a + b - d = 0, \\ &a(d - b) + bd - c^2 = 0, \end{aligned}$$

and the constraint on variables a, b, c, d is given by

$$0 < a < b < c < d.$$

4-, 5- and 6-bit solutions for integer transforms $\frac{1}{\sqrt{q}}$ ISST_8 (a, b, c, d) are shown in Table 5.11. You can note that some integer solutions are multiples of the corresponding unique solutions.

Similarly, the matrix $\tilde{S}_8^{\mathrm{I}}$ does not have a recursive structure, however, it is an EOT. Let $\hat{S}_8^{\mathrm{ISST}}$ be the approximated matrix with its rows in bit-reversed order. In order to get an efficient implementation of $\frac{1}{\sqrt{q}}$ ISST_8 (a, b, c, d), called the fast ISST_8 (a, b, c, d), according to (4.17), and (4.37), (4.38), the approximated matrix $\hat{S}_8^{\mathrm{ISST}}$ can be factorized into the

Table 5.11. 4-, 5- and 6-bit solutions for integer transforms $\frac{1}{\sqrt{q}}$ ISST_8 (a, b, c, d).

a	b	c	d	q
3	5	7	8	294
5	16	19	21	2166
6	10	14	16	1176
7	8	13	15	1014
7	33	37	40	8214
9	15	21	24	2646
10	32	38	42	8664
11	24	31	35	5766
12	20	28	32	4704
13	35	43	48	1094
14	16	26	30	4056
15	25	35	40	7350
15	48	57	63	19494
16	39	49	55	14406
18	30	42	48	10584
21	24	39	45	9126
21	35	49	56	14406
28	32	52	60	16224

following nonrecursive matrix product:

$$S_8^{\mathrm{ISCT}} = \frac{1}{\sqrt{q}} \begin{pmatrix} a & b & c & d & & & & \\ d & -a & -c & b & & 0 & & \\ c & c & 0 & -c & & & & \\ b & -d & c & -a & & & & \\ & & & & a & c & d & b \\ & & & & c & 0 & -c & c \\ & & 0 & & -b & -c & a & d \\ & & & & -d & c & -b & a \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}.$$

5.4.3 Generalized Chen transform

RICOH® Research Center developed a parametrized orthogonal/orthonormal transform (the proprietary transform technology), called the generalized Chen transform (GCT), which is essentially a rationalized approximation of the DCT-II matrix [39, 40–42]. The GCT is based on the parametrization of recursive sparse matrix factorization of the DCT-II matrix [17]. The method exploits the recursive structure of matrix C_N^{II} and relations among its cosine/sine elements. The parametrized GCT can approximate the DCT-II with arbitrary accuracy if the parameters are approximated by dyadic rational numbers (see Section 5.4.4). Consequently, this fact reduces the computational complexity of DCT-II leading to its multiply-free implementation if the normalization is combined with the quantization/dequantization step of transform-based coding scheme. Thus, the GCT is comparable with the DCT-II and for properly chosen approximated parameters it complies with recommendations of international coding standards in terms of accuracy requirements [41].

In the following we present a detailed description of the procedure to derive the parametrized GCT for $N = 8$. Let $\hat{C}_8^{\mathrm{II}}$ be the DCT-II matrix of order 8 with its rows in bit-reversed order. Then, according to equation (4.50) it can be recursively factorized as

$$\begin{aligned} \hat{C}_8^{\mathrm{II}} &= \frac{1}{2} \begin{pmatrix} \hat{C}_4^{\mathrm{II}} & 0 \\ 0 & \hat{C}_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix} \\ &= \frac{1}{2} \begin{pmatrix} \begin{pmatrix} C_1^{\mathrm{II}} & 0 \\ 0 & C_1^{\mathrm{IV}} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} & 0 \\ 0 & C_2^{\mathrm{IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} \begin{pmatrix} 0 & 0 \\ 0 & \hat{C}_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \end{aligned} \tag{5.83}$$

where C_1^{II}, C_2^{II}, C_4^{II} and C_1^{IV}, C_2^{IV}, C_4^{IV} are DCT-II and DCT-IV matrices of order 1, 2 and 4, respectively, and they are given in explicit form as

$$C_1^{\mathrm{II}} = C_1^{\mathrm{IV}} = \frac{1}{\sqrt{2}},$$

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix}, \quad C_2^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \\ \cos\frac{3\pi}{8} & -\sin\frac{3\pi}{8} \end{pmatrix},$$

$$C_4^{\mathrm{II}} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} & -\cos\frac{3\pi}{8} & -\sin\frac{3\pi}{8} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \cos\frac{3\pi}{8} & -\sin\frac{3\pi}{8} & \sin\frac{3\pi}{8} & -\cos\frac{3\pi}{8} \end{pmatrix},$$

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \sin\frac{\pi}{16} \\ \sin\frac{5\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \sin\frac{5\pi}{16} \\ \sin\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} \sin\frac{7\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{7\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix} = \begin{pmatrix} \sin\frac{7\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} \\ \sin\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \sin\frac{5\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix}, \tag{5.84}$$

where cosine/sine elements of C_2^{IV} are related as

$$\cos\frac{\pi}{8} = \sin\frac{3\pi}{8}, \quad \sin\frac{\pi}{8} = \cos\frac{3\pi}{8}, \tag{5.85}$$

while cosine/sine elements of C_4^{IV} are related as

$$\begin{aligned}
\cos\frac{\pi}{16} &= \frac{1}{\sqrt{2}}\left(\cos\frac{3\pi}{16} + \sin\frac{3\pi}{16}\right) = \frac{1}{\sqrt{2}}\left(\sin\frac{5\pi}{16} + \cos\frac{5\pi}{16}\right) = \sin\frac{7\pi}{16}, \\
\sin\frac{\pi}{16} &= \frac{1}{\sqrt{2}}\left(\cos\frac{3\pi}{16} - \sin\frac{3\pi}{16}\right) = \frac{1}{\sqrt{2}}\left(\sin\frac{5\pi}{16} - \cos\frac{5\pi}{16}\right) = \cos\frac{7\pi}{16}, \\
\cos\frac{3\pi}{16} &= \frac{1}{\sqrt{2}}\left(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}\right) = \frac{1}{\sqrt{2}}\left(\sin\frac{7\pi}{16} + \cos\frac{7\pi}{16}\right) = \sin\frac{5\pi}{16}, \\
\sin\frac{3\pi}{16} &= \frac{1}{\sqrt{2}}\left(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}\right) = \frac{1}{\sqrt{2}}\left(\sin\frac{7\pi}{16} - \cos\frac{7\pi}{16}\right) = \cos\frac{5\pi}{16}.
\end{aligned} \tag{5.86}$$

Therefore, in (5.84) we have two different equivalent forms of the matrix C_2^{IV} (and hence also two different equivalent forms of the matrix C_4^{II}), and four different equivalent forms of the matrix C_4^{IV} resulting in totally eight different equivalent forms of the matrix C_8^{II}.

In order to obtain the parametrized version of transform matrix C_8^{II}, the GCT_8 transform, we will parametrize at first the matrix C_2^{IV} and then C_4^{IV}. The parametrized matrices C_2^{IV} and C_4^{IV} will be derived in the form of matrix product $D_M\ R_M$, $M = 2$ and 4, where D_M is a diagonal matrix required for normalization and R_M is a parametrized DCT-IV matrix. Using the trigonometric identities

$$\cos\alpha = \frac{1}{\sqrt{1+\tan^2\alpha}}, \quad \sin\alpha = \frac{\tan\alpha}{\sqrt{1+\tan^2\alpha}}, \tag{5.87}$$

the first form of C_2^{IV} in (5.84) can be parametrized as follows:

$$\begin{aligned} C_2^{\mathrm{IV}} &= \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{8} & 0 \\ 0 & \cos\frac{\pi}{8} \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\pi}{8} \\ \tan\frac{\pi}{8} & -1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\sqrt{1+b^2}} & 0 \\ 0 & \frac{1}{\sqrt{1+b^2}} \end{pmatrix} \begin{pmatrix} 1 & b \\ b & -1 \end{pmatrix}, \end{aligned} \tag{5.88}$$

where $b = \tan\frac{\pi}{8} = \sqrt{2}-1$. Similarly, for the second form of C_2^{IV} in (5.84) we get

$$\begin{aligned} C_2^{\mathrm{IV}} &= \begin{pmatrix} \sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \\ \cos\frac{3\pi}{8} & -\sin\frac{3\pi}{8} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{8} & 0 \\ 0 & \cos\frac{3\pi}{8} \end{pmatrix} \begin{pmatrix} \tan\frac{3\pi}{8} & 1 \\ 1 & -\tan\frac{3\pi}{8} \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\sqrt{1+b^2}} & 0 \\ 0 & \frac{1}{\sqrt{1+b^2}} \end{pmatrix} \begin{pmatrix} b & 1 \\ 1 & -b \end{pmatrix}, \end{aligned} \tag{5.89}$$

where $b = \tan\frac{3\pi}{8} = \sqrt{2}+1$. Thus, we have two different parametrized C_2^{IV} matrices and we denote them as GCT_2-IV (b), where $b = \tan\frac{\pi}{8}$ or $\tan\frac{3\pi}{8}$.

Four equivalent forms of the matrix C_4^{IV} in (5.84) can be parametrized either directly or indirectly through their sparse matrix factorizations. Consider the first explicit form of C_4^{IV} in (5.84). Using the relations among cosine/sine elements given by (5.86) and trigonometric

identities (5.87) the matrix C_4^{IV} can be directly parametrized as

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} \cos\frac{\pi}{16} & & & \\ & \cos\frac{3\pi}{16} & & \\ & & \cos\frac{3\pi}{16} & \\ & & & \cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & \frac{1}{\sqrt{2}}(1+\tan\frac{\pi}{16}) & \frac{1}{\sqrt{2}}(1-\tan\frac{\pi}{16}) & \tan\frac{\pi}{16} \\ 1 & -\frac{1}{\sqrt{2}}(1-\tan\frac{3\pi}{16}) & -\frac{1}{\sqrt{2}}(1+\tan\frac{3\pi}{16}) & -\tan\frac{3\pi}{16} \\ \tan\frac{3\pi}{16} & -\frac{1}{\sqrt{2}}(1+\tan\frac{3\pi}{16}) & \frac{1}{\sqrt{2}}(1-\tan\frac{3\pi}{16}) & 1 \\ \tan\frac{\pi}{16} & -\frac{1}{\sqrt{2}}(1-\tan\frac{\pi}{16}) & \frac{1}{\sqrt{2}}(1+\tan\frac{\pi}{16}) & -1 \end{pmatrix}$$

$$= \begin{pmatrix} \frac{1}{\sqrt{1+a^2}} & & & \\ & \frac{1}{\sqrt{1+c^2}} & & \\ & & \frac{1}{\sqrt{1+c^2}} & \\ & & & \frac{1}{\sqrt{1+a^2}} \end{pmatrix} \begin{pmatrix} 1 & r(1+a) & r(1-a) & a \\ 1 & -r(1-c) & -r(1+c) & -c \\ c & -r(1+c) & r(1-c) & 1 \\ a & -r(1-a) & r(1+a) & -1 \end{pmatrix}, \tag{5.90}$$

where $a = \tan\frac{\pi}{16}$, $c = \tan\frac{3\pi}{16}$ and $r = \frac{1}{\sqrt{2}}$. Denote the parametrized transform matrix C_4^{IV} as GCT_4-IV (a, c, r). Since the reordered matrix $\hat{C}_4^{\mathrm{IV}} J_4$ has the sparse matrix factorization

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{3\pi}{16} \\ -\sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \end{pmatrix} \tag{5.91}$$

$$= \begin{pmatrix} \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix},$$

following the same procedure we can alternatively parametrize the sparse matrix factorization (5.91) into equivalent form as

$$\hat{C}_4^{\mathrm{IV}} J_4 = D_4 \begin{pmatrix} a & 0 & 0 & 1 \\ 0 & 1 & c & 0 \\ 0 & -c & 1 & 0 \\ -1 & 0 & 0 & a \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & r & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} a & r(1-a) & r(1+a) & 1 \\ 1 & r(1-c) & -r(1+c) & c \\ -c & -r(1+c) & -r(1-c) & 1 \\ -1 & r(1+a) & -r(1-a) & a \end{pmatrix}, \tag{5.92}$$

where $D_4 = \text{diag}\,\{\frac{1}{\sqrt{1+a^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+a^2}}\}$ and $a = \tan\frac{\pi}{16}$, $c = \tan\frac{3\pi}{16}$, $r = \frac{1}{\sqrt{2}}$.

Finally, substituting the parametrized DCT-IV matrices $C_2^{\text{IV}} J_2$ given by (5.88) or (5.89) into (5.83) we obtain respectively

$$\hat{C}_8^{\text{II}} = \frac{1}{2}\hat{D}_8 \begin{pmatrix} \begin{pmatrix} 1 & 1 & & 0 \\ 1 & -1 & & \\ & & b & 1 \\ 0 & & -1 & b \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \hat{C}_4^{\text{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.93}$$

where $b = \tan\frac{\pi}{8}$, or

$$\hat{C}_8^{\text{II}} = \frac{1}{2}\hat{D}_8 \begin{pmatrix} \begin{pmatrix} 1 & 1 & & 0 \\ 1 & -1 & & \\ & & 1 & b \\ 0 & & -b & 1 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \hat{C}_4^{\text{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.94}$$

where $b = \tan\frac{3\pi}{8}$, $\hat{D}_8 = \text{diag}\,\{\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}}, \frac{1}{\sqrt{1+b^2}}, \frac{1}{\sqrt{1+b^2}}, \frac{1}{\sqrt{1+a^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+a^2}}\}$ and $\hat{C}_4^{\text{IV}} J_4$ is given by (5.92).

Equations (5.93) and (5.94) lead to the following important conclusions:

- They define the parametrized GCT_8 or two equivalent parametrized C_8^{II} matrices which are given respectively as

$$C_8^{\text{II}} = D_8 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & r(1+a) & r(1-a) & a & -a & -r(1-a) & -r(1+a) & -1 \\ 1 & b & -b & -1 & -1 & -b & b & 1 \\ 1 & -r(1-c) & -r(1+c) & -c & c & r(1+c) & r(1-c) & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ c & -r(1+c) & r(1-c) & 1 & -1 & -r(1-c) & r(1+c) & -c \\ b & -1 & 1 & -b & -b & 1 & -1 & b \\ a & -r(1-a) & r(1+a) & -1 & 1 & -r(1+a) & r(1-a) & -a \end{pmatrix}, \tag{5.95}$$

where $b = \tan\frac{\pi}{8}$, and

$$C_8^{\mathrm{II}} = D_8 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & r(1+a) & r(1-a) & a & -a & -r(1-a) & -r(1+a) & -1 \\ b & 1 & -1 & -b & -b & -1 & 1 & b \\ 1 & -r(1-c) & -r(1+c) & -c & c & r(1+c) & r(1-c) & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ c & -r(1+c) & r(1-c) & 1 & -1 & -r(1-c) & r(1+c) & -c \\ 1 & -b & b & -1 & -1 & b & -b & 1 \\ a & -r(1-a) & r(1+a) & -1 & 1 & -r(1+a) & r(1-a) & -a \end{pmatrix}, \tag{5.96}$$

where $b = \tan\frac{3\pi}{8}$, and $D_8 = \mathrm{diag}\{\frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{1+a^2}}, \frac{1}{2\sqrt{1+b^2}}, \frac{1}{2\sqrt{1+c^2}}, \frac{1}{2\sqrt{2}}, \frac{1}{2\sqrt{1+c^2}}, \frac{1}{2\sqrt{1+b^2}}, \frac{1}{2\sqrt{1+a^2}}\}$. For the set of real-valued parameters $a = \tan\frac{\pi}{16}$, $b = \tan\frac{\pi}{8}$ or $\tan\frac{3\pi}{8}$, $c = \tan\frac{3\pi}{16}$ and $r = \frac{1}{\sqrt{2}}$ the parametrized GCT_8 denoted as GCT_8-II (a, b, c, r) is exactly the ordinary DCT-II matrix C_8^{II}.

- They define implicitly lower-order parametrized transform matrices $C_2^{\mathrm{II}}, C_2^{\mathrm{IV}}, C_4^{\mathrm{II}}, C_4^{\mathrm{IV}}$ denoted as GCT_2-II (r), GCT_2-IV (b), GCT_4-II (b, r), GCT_4-IV (a, c, r), respectively.
- They define together with the sparse matrix factorization of $\hat{C}_4^{\mathrm{IV}} J_4$ given by (5.92) the fast GCT_8-II (a, b, c, r).
- In general, the higher-order parametrized GCT of order $2N$ can be generated from lower-order parametrized matrices C_N^{II} and C_N^{IV}. All that is necessary is to parametrize the matrix C_N^{IV} and derive its sparse matrix factorization to complete its efficient implementation.

For completeness we present the parametrizations of the remaining three equivalent forms of the matrix C_4^{IV} (direct parametrization and parametrization through its sparse matrix factorization). Substituting them into (5.93) and (5.94) the remaining six forms of parametrized GCT_8-II (a, b, c, r) can be obtained. In all cases the diagonal matrix is given by $D_4 = \mathrm{diag}\{\frac{1}{\sqrt{1+a^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+c^2}}, \frac{1}{\sqrt{1+a^2}}\}$.

(A) GCT_4-IV (a,c,r), $a = \tan\frac{\pi}{16}$, $c = \tan\frac{5\pi}{16}$, $r = \frac{1}{\sqrt{2}}$

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \sin\frac{\pi}{16} \\ \sin\frac{5\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \sin\frac{5\pi}{16} \\ \sin\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix} = D_4 \begin{pmatrix} 1 & r(1+a) & r(1-a) & a \\ c & -r(c-1) & -r(c+1) & -1 \\ 1 & -r(c+1) & r(c-1) & c \\ a & -r(1-a) & r(1+a) & -1 \end{pmatrix},$$

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \sin\frac{\pi}{16} & \cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{\pi}{16} \\ \sin\frac{5\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \cos\frac{5\pi}{16} \\ -\cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \sin\frac{5\pi}{16} \\ -\cos\frac{\pi}{16} & \sin\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ 0 & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & 0 \\ -\cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= D_4 \begin{pmatrix} a & 0 & 0 & 1 \\ 0 & c & 1 & 0 \\ 0 & -1 & c & 0 \\ -1 & 0 & 0 & a \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & r & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} a & r(1-a) & r(1+a) & 1 \\ c & r(c-1) & -r(c+1) & c \\ -1 & -r(c+1) & -r(c-1) & c \\ -1 & r(1+a) & -r(1-a) & a \end{pmatrix}. \tag{5.97}$$

(B) GCT$_4$-IV (a, c, r), $\mathbf{a = \tan\frac{7\pi}{16}}$, $\mathbf{c = \tan\frac{3\pi}{16}}$, $\mathbf{r = \frac{1}{\sqrt{2}}}$

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \sin\frac{7\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{7\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix} = D_4 \begin{pmatrix} a & r(a+1) & r(a-1) & 1 \\ 1 & -r(1-c) & -r(1+c) & -c \\ c & -r(1+c) & r(1-c) & 1 \\ 1 & -r(a-1) & r(a+1) & -a \end{pmatrix},$$

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \cos\frac{7\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & \sin\frac{3\pi}{16} \\ -\sin\frac{3\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} \\ -\sin\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{7\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= D_4 \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & c & 0 \\ 0 & -c & 1 & 0 \\ -a & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & r & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & r(a-1) & r(a+1) & a \\ 1 & r(1-c) & -r(1+c) & c \\ -c & -r(1+c) & -r(1-c) & 1 \\ -a & r(a+1) & -r(a-1) & 1 \end{pmatrix}. \tag{5.98}$$

(C) GCT$_4$-IV(a,c,r), $a = \tan\frac{7\pi}{16}$, $c = \tan\frac{5\pi}{16}$, $r = \frac{1}{\sqrt{2}}$

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \sin\frac{7\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} \\ \sin\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \sin\frac{5\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix} = D_4 \begin{pmatrix} a & r(a+1) & r(a-1) & 1 \\ c & -r(c-1) & -r(c+1) & -1 \\ 1 & -r(c+1) & r(c-1) & c \\ 1 & -r(a-1) & r(a+1) & -a \end{pmatrix},$$

$$\hat{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \cos\frac{7\pi}{16} & \cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & \sin\frac{7\pi}{16} \\ \sin\frac{5\pi}{16} & \cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{5\pi}{16} \\ -\cos\frac{5\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \sin\frac{5\pi}{16} \\ -\sin\frac{7\pi}{16} & \sin\frac{5\pi}{16} & -\cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} \end{pmatrix}$$

$$= \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ 0 & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= D_4 \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & c & 1 & 0 \\ 0 & -1 & c & 0 \\ -a & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -r & r & 0 \\ 0 & r & r & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$= \begin{pmatrix} 1 & r(a-1) & r(a+1) & a \\ c & r(c-1) & -r(c+1) & 1 \\ -1 & -r(c+1) & -r(c-1) & c \\ -a & r(a+1) & -r(a-1) & 1 \end{pmatrix}. \tag{5.99}$$

Note: There exist even better parametrizations of four different equivalent forms of the matrix C_4^{IV} in terms of the number of parameters including two parameters only. Unfortunately, sparse matrix factorizations of these parametrized forms of C_4^{IV} do not exist. These four parametrized C_4^{IV} matrices with two parameters denoted by GCT_4-IV (a, r) are respectively given as

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix}$$

$$= \frac{1}{\sqrt{1+a^2}} \begin{pmatrix} 1 & r(a+1) & r(1-a) & a \\ r(1+a) & -a & -1 & -r(1-a) \\ r(1-a) & -1 & a & r(1+a) \\ a & -r(1-a) & r(1+a) & -1 \end{pmatrix}, \quad a = \tan\frac{\pi}{16}, \quad r = \frac{1}{\sqrt{2}},$$

•

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \cos\frac{\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \sin\frac{\pi}{16} \\ \sin\frac{5\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \sin\frac{5\pi}{16} \\ \sin\frac{\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix}$$

$$= \frac{1}{\sqrt{1+a^2}} \begin{pmatrix} r(1+a) & 1 & a & r(1-a) \\ 1 & -r(1-a) & -r(1+a) & -a \\ a & -r(1+a) & r(1-a) & 1 \\ r(1-a) & -a & 1 & -r(1+a) \end{pmatrix}, \quad a = \tan\frac{3\pi}{16}, \quad r = \frac{1}{\sqrt{2}},$$

•

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \sin\frac{7\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{7\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix}$$

$$= \frac{1}{\sqrt{1+a^2}} \begin{pmatrix} r(a+1) & a & 1 & r(a-1) \\ a & -r(a-1) & -r(a+1) & -1 \\ 1 & -r(a+1) & r(a-1) & a \\ r(a-1) & -1 & a & -r(a+1) \end{pmatrix}, \quad a = \tan\frac{5\pi}{16}, \quad r = \frac{1}{\sqrt{2}},$$

•

$$C_4^{\mathrm{IV}} = \begin{pmatrix} \sin\frac{7\pi}{16} & \sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & \cos\frac{7\pi}{16} \\ \sin\frac{5\pi}{16} & -\cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{5\pi}{16} \\ \cos\frac{5\pi}{16} & -\sin\frac{7\pi}{16} & \cos\frac{7\pi}{16} & \sin\frac{5\pi}{16} \\ \cos\frac{7\pi}{16} & -\cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & -\sin\frac{7\pi}{16} \end{pmatrix}$$

$$= \frac{1}{\sqrt{1+a^2}} \begin{pmatrix} a & r(a+1) & r(a-1) & 1 \\ r(a+1) & -1 & -a & -r(a-1) \\ r(a-1) & -a & 1 & r(a+1) \\ 1 & -r(a-1) & r(a+1) & -a \end{pmatrix}, \quad a = \tan\frac{7\pi}{16}, \quad r = \frac{1}{\sqrt{2}},$$

•

5.4.3.1 The fast GCT$_8$-II (a,b,c,r)

Equations (5.93) and (5.94) with the sparse matrix factorization of $\hat{C}_4^{\mathrm{IV}} J_4$ given by (5.92) define the fast GCT$_8$-II (a, b, c, r) for the efficient implementation of parametrized GCT$_8$-II $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}})$ and GCT$_8$-II $(\tan\frac{\pi}{16}, \tan\frac{3\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}})$, respectively. The corresponding generalized signal flow graphs are shown in Figs. 5.8 and 5.9.

We note that GCT$_8$-II (a, b, c, r) originally proposed in Refs. [39, 40–42] is a combination of GCT$_2$-II $(\frac{1}{\sqrt{2}})$, GCT$_2$-IV $(\tan\frac{3\pi}{16})$ given by (5.89) and GCT$_4$-IV $(\tan\frac{7\pi}{16}, \tan\frac{5\pi}{16}, \frac{1}{\sqrt{2}})$ given by (5.99), all substituted into (5.83).

Each one of the eight equivalent forms of GCT$_8$-II (a, b, c, r) has the associated fast GCT$_8$-II (a, b, c, r). They differ only by appropriate butterfly stages corresponding to GCT$_2$ -IV

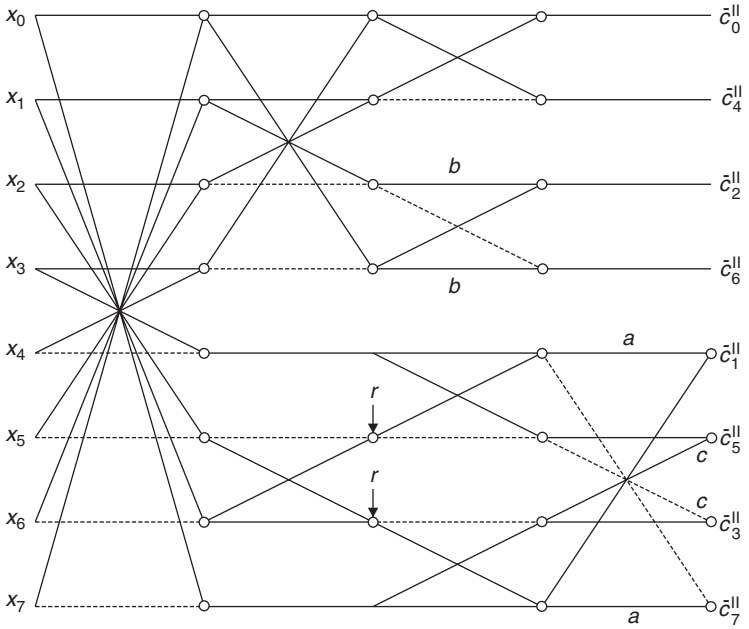


Fig. 5.8. The fast GCT_8-II ($\tan\frac{\pi}{16}$, $\tan\frac{\pi}{8}$, $\tan\frac{3\pi}{16}$, $\frac{1}{\sqrt{2}}$) implementation.

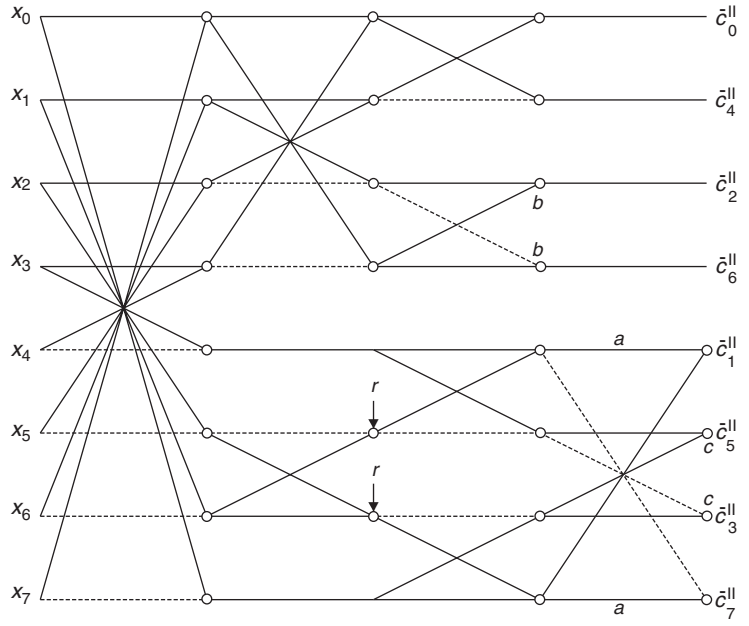


Fig. 5.9. The fast GCT_8-II ($\tan\frac{\pi}{16}$, $\tan\frac{3\pi}{8}$, $\tan\frac{3\pi}{16}$, $\frac{1}{\sqrt{2}}$) implementation.

Table 5.12. Selected approximations of the parameters a, b, c, r by dyadic rationals.

$\tan \frac{\pi}{8} = \sqrt{2} - 1$	**0.414214**
3/8	0.375000
13/32	0.406250
53/128	0.414063
$\tan \frac{3\pi}{8} = \sqrt{2} + 1$	**2.414214**
19/8	2.375000
77/32	2.406250
309/128	2.414063
$\tan \frac{\pi}{16}$	**0.198912**
3/16	0.187500
25/128	0.195313
51/256	0.199219
$\tan \frac{3\pi}{16}$	**0.668179**
11/16	0.687500
85/128	0.664063
171/256	0.667969
$\tan \frac{5\pi}{16}$	**1.496606**
3/2	1.500000
191/128	1.492188
383/256	1.496094
$\tan \frac{7\pi}{16}$	**5.027339**
5	5.000000
161/32	5.031250
643/256	5.023438
$\frac{1}{\sqrt{2}}$	**0.707107**
11/16	0.687500
91/128	0.710938
181/256	0.707031

(b) and GCT$_4$-IV (a, c, r). It is interesting to note that if GCT$_8$-II (a, b, c, r) is viewed as a real-valued transform, then with respect to signal flow graphs in Figs. 5.9 and 5.10 the computation of 8-point DCT-II without normalization requires only 8 multiplications and 26 additions.

At this moment GCT$_8$-II (a, b, c, r) is real-valued with the set of parameters $\{a, b, c, r\}$ which are irrational numbers. However, approximating parameters a, b, c, r by dyadic rationals, i.e., in the form $\frac{k}{2^m}$, where k, m are integers, and k is an odd, results in the final rationalized approximation of the DCT-II matrix C_8^{II}. Rational approximations of the parameters increase the computational speed in implementing the GCT$_8$-II (a, b, c, r) since binary addition and shift operations replacing multiplications enable a multiply-free implementation of the fast GCT$_8$-II (a, b, c, r). The GCT$_8$-II (a, b, c, r) can operate with arbitrarily chosen parameters which can approximate the DCT-II matrix C_8^{II} with arbitrary accuracy. The closer the approximated parameters a, b, c, r are to the original irrational values, the more perfect is the resulting GCT$_8$-II (a, b, c, r), but at a higher computational and/or hardware cost. Thus, the GCT$_8$-*II* (a, b, c, r) is a flexible transform with different

Table 5.13. Comparison of the MSE approximation error and performance measures of rationalized GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$) with the 8-point DCT-II.

GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$)	MSE	C_g	η
8-point DCT-II		8.82591	93.99119
GCT_8-II (3/16, 3/8, 11/16, 91/128)	4.235859e−005	8.79995	93.82347
GCT_8-II (25/128, 13/32, 85/128, 91/128)	3.541383e−006	8.80372	93.89385
GCT_8-II (51/256, 53/128, 171/256, 181/256)	7.783073e−009	8.82635	93.99570

Table 5.14. The computational complexity of rationalized GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$).

GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$)	Adds/shifts
GCT_8-II (3/16, 3/8, 11/16, 11/16)	38/20
GCT_8-II (3/16, 3/8, 11/16, 91/128)	42/24
GCT_8-II (25/128, 13/32, 85/128, 91/128)	48/30
GCT_8-II (51/256, 53/128, 171/256, 181/256)	54/36

desirable characteristics. Rational parameters are chosen as a tradeoff between computational efficiency and desired accuracy. In Table 5.12 are shown selected approximations of parameters a, b, c, r by dyadic rationals.

Comparison of the MSE approximation error and performance measures of some rationalized GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$) with the 8-point DCT-II is summarized in Table 5.13. The computational complexity of rationalized GCT_8-II ($\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16}, \frac{1}{\sqrt{2}}$) with respect to signal flow graphs in Figs. 5.9 and 5.10 is summarized in Table 5.14.

Note: It may seem that if the DCT/DST matrix has a recursive structure and its sparse matrix factorization exists, then it may be parametrizable. This is not the case. Although the DCT-I matrix C^{I}_{N+1} and DST-I matrix S^{I}_{N-1} defined by (4.1) and (4.5), respectively, have the recursive structure they cannot be parametrized and their rationalized approximations do not exist.

5.4.3.2 Generalized CMT

The parametrized GCT motivated us to derive quite different and new parametrized DCT-II matrix C^{II}_8 which we will call the generalized CMT (GCMT). It is based on the computation of DCT-II via WHT (see Section 4.4.3.2) through the conversion matrix which has a sparse-block-diagonal structure and it can be recursively generated from lower-order block matrices. Let $\hat{C}^{\mathrm{II}}_8$ be the DCT-II matrix of order 8 with its rows in bit-reversed order. From (4.60) it follows that the matrix $\hat{C}^{\mathrm{II}}_8$ can be expressed as

$$\hat{C}^{\mathrm{II}}_8 = T_8 \hat{W}_8, \quad T_8 = \hat{C}^{\mathrm{II}}_8 \hat{W}^{\mathrm{T}}_8, \tag{5.100}$$

where $\hat{W}_8^{\mathrm{T}}$ is the sequency ordered WHT matrix and T_8 is the conversion matrix given by

$$T_8 = \begin{pmatrix} 1 & & & 0 \\ & 1 & & \\ & & U_2 & \\ 0 & & & U_4 \end{pmatrix}, \tag{5.101}$$

where the block matrices U_2 and U_4 with the sparse matrix factorization are given by

$$U_2 = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix},$$

$$U_4 = P_4 \begin{pmatrix} \cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} U_2 & 0 \\ 0 & U_2 \end{pmatrix} P_4,$$

and P_4 is a permutation matrix given by

$$P_4 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

In order to obtain the parametrized GCMT it is sufficient to parametrize the conversion matrix T_8, or in other words, to parametrize the block matrices U_2 and U_4. First, let us parametrize the block matrix U_2. Using the trigonometric identity (5.87) we get

$$\begin{aligned} U_2 = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} &= \begin{pmatrix} \cos\frac{\pi}{8} & 0 \\ 0 & \cos\frac{\pi}{8} \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\pi}{8} \\ -\tan\frac{\pi}{8} & 1 \end{pmatrix} \\ &= \begin{pmatrix} \frac{1}{\sqrt{1+b^2}} & 0 \\ 0 & \frac{1}{\sqrt{1+b^2}} \end{pmatrix} \begin{pmatrix} 1 & b \\ -b & 1 \end{pmatrix}, \end{aligned} \tag{5.102}$$

where $b = \tan\frac{\pi}{8}$. Similarly, for the factorized block matrix U_4 we obtain

$$
\begin{aligned}
U_4 = P_4 &\begin{pmatrix} \cos\frac{\pi}{16} & & & \\ & \cos\frac{3\pi}{16} & & \\ & & \cos\frac{3\pi}{16} & \\ & & & \cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & \tan\frac{\pi}{16} \\ 0 & 1 & \tan\frac{3\pi}{16} & 0 \\ 0 & -\tan\frac{3\pi}{16} & 1 & 0 \\ -\tan\frac{\pi}{16} & 0 & 0 & 1 \end{pmatrix} \\
&\times \begin{pmatrix} \cos\frac{\pi}{8} & & & \\ & \cos\frac{\pi}{8} & & \\ & & \cos\frac{\pi}{8} & \\ & & & \cos\frac{\pi}{8} \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\pi}{8} & 0 & 0 \\ -\tan\frac{\pi}{8} & 1 & 0 & \\ 0 & 0 & 1 & \tan\frac{\pi}{8} \\ 0 & 0 & -\tan\frac{\pi}{8} & 1 \end{pmatrix} P_4 \\
&= P_4 \frac{1}{\sqrt{1+b^2}} \begin{pmatrix} \frac{1}{\sqrt{1+a^2}} & & & \\ & \frac{1}{\sqrt{1+c^2}} & & \\ & & \frac{1}{\sqrt{1+c^2}} & \\ & & & \frac{1}{\sqrt{1+a^2}} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & c & 0 \\ 0 & -c & -1 & 1 \\ -a & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & b & 0 & 0 \\ -b & 1 & 0 & 0 \\ 0 & 0 & 1 & b \\ 0 & 0 & -b & 1 \end{pmatrix} P_4,
\end{aligned}
\tag{5.103}
$$

where $a = \tan\frac{\pi}{16}$ and $c = \tan\frac{3\pi}{16}$. Substituting the parametrized block matrices U_2 and U_4 into (5.101) the conversion matrix T_8 has the following form:

$$
\bar{T}_8 = \frac{1}{\sqrt{8}} \hat{D}_8 \begin{pmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & 1 & b & & & & \\ & & -b & 1 & & & & \\ & & & & 1 & -ab & b & a \\ & & & & cb & 1 & -c & b \\ & & & & -b & c & 1 & cb \\ & & & & -a & -b & -ab & 1 \end{pmatrix}, \tag{5.104}
$$

where $\hat{D}_8 = \mathrm{diag}\{1, 1, \frac{1}{\sqrt{1+b^2}}, \frac{1}{\sqrt{1+b^2}}, \frac{1}{\sqrt{(1+a^2)(1+b^2)}}, \frac{1}{\sqrt{(1+c^2)(1+b^2)}}, \frac{1}{\sqrt{(1+c^2)(1+b^2)}}, \frac{1}{\sqrt{(1+a^2)(1+b^2)}}\}$. Finally, according to (5.100) by evaluating the matrix product $T_8\,\hat{W}_8$ and exploiting the EOT factorization (4.17), the parametrized $\hat{C}_8^{\mathrm{II}}$ matrix is given by

$$
\hat{C}_8^{\mathrm{II}} = \frac{1}{\sqrt{8}} \hat{D}_8 \begin{pmatrix} E_4 & 0 \\ 0 & O_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix} \tag{5.105}
$$

where

$$E_4 = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & -1 & 1 \\ 1+b & 1-b & -(1-b) & -(1+b) \\ 1-b & -(1+b) & 1+b & -(1-b) \end{pmatrix},$$

$$O_4 = \begin{pmatrix} -[(1-a)-b(1+a)] & -[(1+a)-b(1-a)] & -[(1-a)+b(1+a)] & -[(1+a)+b(1-a)] \\ -[(1+c)-b(1-c)] & (1-c)-b(1+c) & (1+c)+b(1-c) & -[(1-c)+b(1+c)] \\ (1-c)+b(1+c) & (1+c)+b(1-c) & -[(1-c)-b(1+c)] & -[(1+c)-b(1-c)] \\ (1+a)+b(1-a) & -[(1-a)+b(1+a)] & (1+a)-b(1-a) & -[(1-a)-b(1+a)] \end{pmatrix}.$$

5.4.3.3 The fast GCMT$_8$ (a, b, c)

Let us denote the new parametrized transform as GCMT$_8$ (a, b, c), where $a = \tan\frac{\pi}{16}$, $b = \tan\frac{\pi}{8}$ and $c = \tan\frac{3\pi}{16}$. Equation (5.100) and the parametrized block matrices U_2 and U_4 in (5.102) and (5.103) define the fast GCMT$_8$ (a, b, c). The corresponding generalized signal flow graph is shown in Fig. 5.10. Again, it is interesting to note that if GCMT$_8$ (a, b, c) is viewed as the actual transform, then with respect to the signal flow graph in Fig. 5.10 the computation of 8-point DCT-II without normalization requires only 10 multiplications and 34 additions.

Approximating the set of parameters $\{a, b, c\}$ by dyadic rationals (see Table 5.12) we obtain new rationalized approximation of the DCT-II matrix C_8^{II}. Compared to the integer CMT described in Section 5.4.1, the GCMT$_8$ (a, b, c) is more elegant and flexible with multiply-free implementation. In the Table 5.15, there is a summarized comparison of the MSE approximation error and performance measures of some rationalized GCMT$_8$ $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$ with the 8-point DCT-II. The multiply-free computational complexity of rationalized GCMT$_8$ $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$ with respect to the signal flow graphs in Fig. 5.10 is summarized in Table 5.16.

5.4.4 BinDCT/BinDST and IntDCT/IntDST

Fast multiplierless approximations of the 8-point DCT-II, called the BinDCT (Binary arithmetic DCT) [43–45, 46, 50, 51] and IntDCT (Integer DCT) [47, 50, 51] represent modern transform technologies to approximation of the DCT-II in integer domain. Although the BinDCT and IntDCT have been proposed independently, the simple and elegant methods for their construction are the same. They differ only in the fast DCT-II algorithm employed for the approximation.

In the theory of fast algorithms for the discrete orthogonal transforms computation, a recursive or nonrecursive sparse matrix factorization of a transform matrix defines a fast algorithm which is represented by the corresponding generalized signal flow graph and vice versa. In many cases the sparse matrix factorization of DCT-II matrix consists of the product of butterfly matrices whose elements are ± 1, and rotation matrices composed of cascades of 2×2 plane rotations (see Section 4.4.3). Invertible BinDCTs and IntDCTs

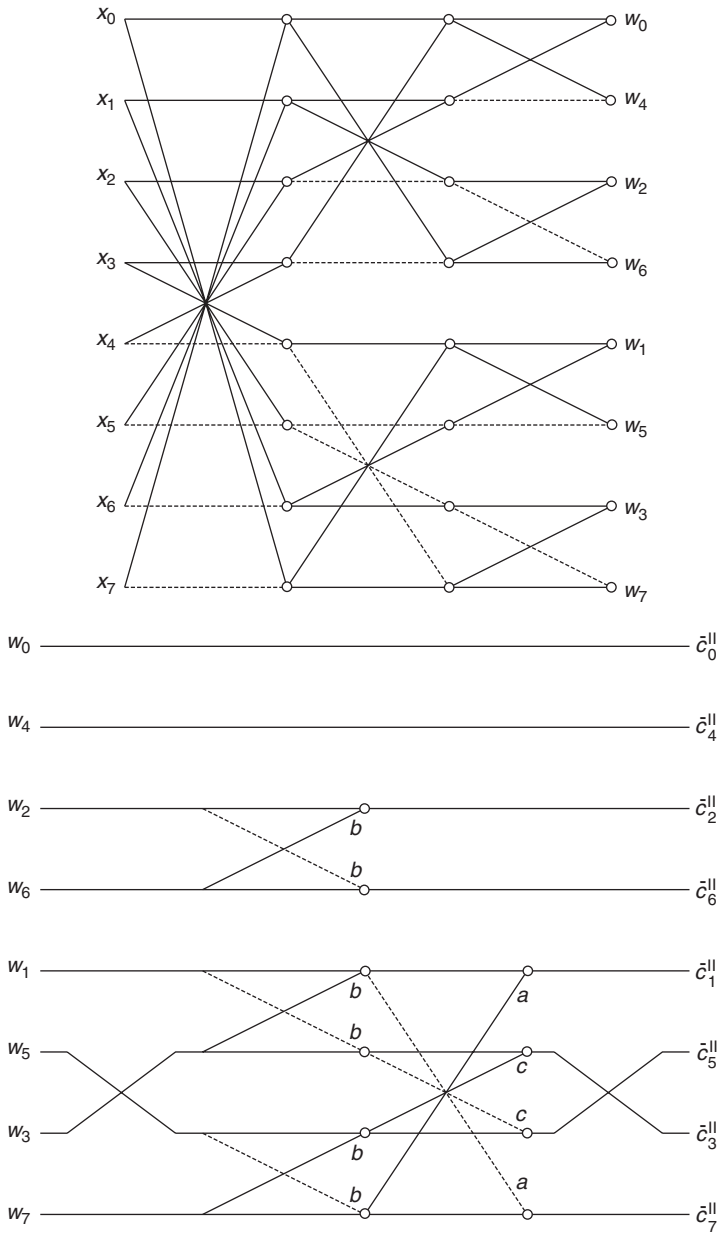


Fig. 5.10. The fast GCMT$_8$ (a, b, c) implementation.

are derived from plane rotation-based sparse matrix factorizations of the DCT-II matrix exploiting matrix factorizations of Givens–Jacobi rotations G_φ and Householder reflections H_φ. In Section 5.2 it was shown that G_φ and H_φ can be factored into product of Gauss–Jordan elementary matrices being the unit lower and unit upper triangular matrices, and a diagonal matrix. Each matrix factorization of G_φ (LUL, ULU, DLU, DUL)

Table 5.15. Comparison of the MSE approximation error and performance measures of rationalized GCMT_8 $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$ with the 8-point DCT-II.

GCMT_8 $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$	MSE	C_g	η
8-point DCT-II		8.82591	93.99119
GCMT_8 (3/16, 3/8, 11/16)	1.428576e−004	8.81701	93.49390
GCMT_8 (25/128, 13/32, 85/128)	6.380515e−006	8.82571	93.93420
GCMT_8 (51/256, 53/128, 171/256)	8.885787e−009	8.82590	93.98777

Table 5.16. The computational complexity of rationalized GCMT_8 $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$.

GCMT_8 $(\tan\frac{\pi}{16}, \tan\frac{\pi}{8}, \tan\frac{3\pi}{16})$	Adds/shifts
GCMT_8 (3/16, 3/8, 11/16)	46/22
GCMT_8 (25/128, 13/32, 85/128)	56/32
GCMT_8 (51/256, 53/128, 171/256)	54/36

and factorization of H_φ (DLU, DUL) has an associated efficient computational structure for the forward and inverse computation. For DCT-II, factorizing G_φ and H_φ is just another way to implement the conventional floating-point DCT-II. Matrix factorizations of G_φ and H_φ offer the versatility in constructing multiplierless approximations to fast transforms.

The multiplierless approximation of a fast transform is obtained by replacing each 2×2 plane rotation in the sparse matrix factorization of the transform matrix (or equivalently in the generalized signal flow graph) by its proper LUL, ULU, DLU or DUL matrix factorization (or equivalently by its corresponding efficient computational structure), and approximating the floating-point multipliers with dyadic rationals. Dyadic approximations of multipliers provide fast, efficient in-place computation of transform coefficients, the ability to map integers to integers with perfect reconstruction property and an elegant implementation utilizing only add and shift operations. The resulting BinDCTs and IntDCTs closely approximate the floating-point DCT-II. They can have different computational complexities depending on the required approximation precision. In general, with a higher computational complexity a higher accuracy is achieved. Thus, the various configurations of BinDCT and IntDCT can be tailored to the requirements of a given application.

The method can be straightforwardly extended to any DCT and DST [52, 54, 56], and generally to any discrete sinusoidal transform such as the generalized DHL (GDHT) or the generalized discrete W transform (GDWT) [55, 57] as long as a plane rotation-based sparse matrix factorization of the transform matrix is known. Actually, such factorizations for all DCTs and DSTs exist and they are presented in Section 4.4. Since complex multiplications in FFT algorithms by twiddle factors $e^{j\varphi}$ can be converted to the Givens–Jacobi rotations of complex vector (see equation (5.15)), the method has been applied to split-radix FFT

algorithm [14–16] to construct fast multiplierless approximation of the DFT called Integer FFT (IntFFT) [48, 49, 53].

5.4.4.1 Matrix factorizations of G_{φ_i} and H_{φ_i}: notations and analysis

Matrix factorizations of Givens–Jacobi rotations G_{φ_i} and Householder reflections H_{φ_i} are key mathematical tools in constructing the multiplierless approximations of fast transforms. In Section 5.2 we showed that G_{φ_i} defined by (5.13) and H_{φ_i} defined by (5.14) can be factored into the product of Gauss–Jordan elementary matrices that are unit lower and unit upper triangular matrices, and a diagonal matrix. LUL, ULU, DLU (ULD) and DUL (LUD) efficient computational structures of G_{φ_i} (forward and inverse) which correspond to LUL, ULU, DLU (ULD) and DUL (LUD) factorizations are respectively shown in Figs. 5.1–5.4. DLU and DUL structures of H_{φ_i} are quite similar to that of G_{φ_i} except for sign changes. We note that LUL and ULU factorizations (and hence their efficient structures) are equivalent in view of the realization of computation, and they are alternatives to each other. Therefore, the choice of LUL or ULU structure for the efficient implementation of G_{φ_i} in practical applications is left to the reader. Similar conclusion is valid for DLU and DUL structures of G_{φ_i} and H_{φ_i}. We recall that there exist relationships between G_{φ_i} and H_{φ_i} given by (5.36) and (5.37), and the structure of G_{φ_i} can be adapted with minor modification for the implementation of H_{φ_i} and vice versa. Furthermore, any other plane rotation R_{φ_i} (see Section 5.2.8) can be always converted to G_{φ_i}.

In this section, we prefer LUL and DLU factorizations of G_{φ_i} with the corresponding computational structures and in the following they are discussed in detail. The LUL factorizations of G_{φ_i} and its inverse, $G_{-\varphi_i}$, with the multipliers p_i and u_i are respectively defined as

$$G_{\varphi_i} = \begin{pmatrix} \cos\varphi_i & -\sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ p_i & 1 \end{pmatrix} \begin{pmatrix} 1 & -u_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ p_i & 1 \end{pmatrix},$$

$$G_{-\varphi_i} = \begin{pmatrix} \cos\varphi_i & \sin\varphi_i \\ -\sin\varphi_i & \cos\varphi_i \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -p_i & 1 \end{pmatrix} \begin{pmatrix} 1 & u_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -p_i & 1 \end{pmatrix}, \tag{5.106}$$

where

$$p_i = \frac{1-\cos\varphi_i}{\sin\varphi_i} = \tan\frac{\varphi_i}{2} \quad \text{and} \quad u_i = \sin\varphi_i \neq 0.$$

The corresponding forward and inverse LUL structures are respectively shown in Fig. 5.11(a) and (b), where $(x_0, x_1)^{\mathrm{T}}$ and $(y_0, y_1)^{\mathrm{T}}$ are the input and rotated vectors, respectively.

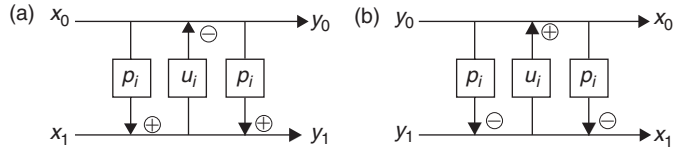


Fig. 5.11. The LUL structures: (a) of G_{φ_i} and (b) of its inverse $G_{-\varphi_i}$.

In the practical implementations of the LUL structure of G_{φ_i} for a rotation angle φ_i the value of the multiplier u_i can be arbitrarily small and consequently, the dynamic range of multiplier p_i can be arbitrarily large. Therefore, we should control the range of the multipliers p_i and u_i as follows. If the rotation angle $\varphi_i \in (-\frac{\pi}{2}, 0) \cup (0, \frac{\pi}{2})$, then the value of $\cos\varphi_i > 0$ and

$$|p_i| = \left|\tan\frac{\varphi_i}{2}\right| = \left|\frac{1-\cos\varphi_i}{\sin\varphi_i}\right| = \left|\frac{\sin\varphi_i}{1+\cos\varphi_i}\right|$$
$$= |\sin\varphi_i| \left|\frac{1}{1+\cos\varphi_i}\right| < |\sin\varphi_i| = |u_i| < 1, \tag{5.107}$$

and multipliers p_i and u_i fall between -1 and $+1$. However, if the rotation angle $\varphi_i \in (-\pi, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$, then the value of $\cos\varphi_i < 0$ and $|p_i| > 1$ which is not desirable. The absolute values of multipliers p_i and u_i can be controlled to be always less than or equal to 1 by replacing G_{φ_i} with its equivalent form $-G_{\varphi_i+\pi}$ as [49]

$$G_{\varphi_i} = -G_{\varphi_i+\pi} = -\begin{pmatrix} -\cos\varphi_i & \sin\varphi_i \\ -\sin\varphi_i & -\cos\varphi_i \end{pmatrix} = -\begin{pmatrix} 1 & 0 \\ \bar{p}_i & 1 \end{pmatrix}\begin{pmatrix} 1 & -u_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ \bar{p}_i & 1 \end{pmatrix}, \tag{5.108}$$

where

$$\bar{p}_i = \frac{1+\cos\varphi_i}{\sin\varphi_i} = \frac{1}{\tan\frac{\varphi_i}{2}} \quad \text{and} \quad u_i = \sin\varphi_i \neq 0.$$

When rotation angle $\varphi_i \in (-\pi, \frac{\pi}{2}) \cup (\frac{\pi}{2}, \pi)$, then we have

$$|\bar{p}_i| = \left|\frac{1}{\tan\frac{\varphi_i}{2}}\right| = \left|\frac{1+\cos\varphi_i}{\sin\varphi_i}\right| = \left|\frac{\sin\varphi_i}{1-\cos\varphi_i}\right|$$
$$= |\sin\varphi_i| \left|\frac{1}{1-\cos\varphi_i}\right| < |\sin\varphi_i| = |u_i| < 1, \tag{5.109}$$

and new multipliers $\bar{p}_i$ and u_i fall again between -1 and $+1$. In order to obtain the corresponding LUL structure of $-G_{\varphi_i+\pi}$ we need to modify the LUL structure of G_{φ_i} replacing the multiplier p_i by $\bar{p}_i$ and changing signs of the result.

Now consider the DLU and ULD factorizations of G_{φ_i} and its inverse, $G_{\varphi_i}^{-1}$, respectively, with the multipliers p_i and u_i which are defined as

$$G_{\varphi_i} = \begin{pmatrix} \cos\varphi_i & -\sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i \end{pmatrix} = \begin{pmatrix} \cos\varphi_i & 0 \\ 0 & \frac{1}{\cos\varphi_i} \end{pmatrix}\begin{pmatrix} 1 & 0 \\ u_i & 1 \end{pmatrix}\begin{pmatrix} 1 & -p_i \\ 0 & 1 \end{pmatrix},$$

$$G_{\varphi_i}^{-1} = \begin{pmatrix} \cos\varphi_i & \sin\varphi_i \\ -\sin\varphi_i & \cos\varphi_i \end{pmatrix} = \begin{pmatrix} 1 & p_i \\ 0 & 1 \end{pmatrix}\begin{pmatrix} 1 & 0 \\ -u_i & 1 \end{pmatrix}\begin{pmatrix} \frac{1}{\cos\varphi_i} & 0 \\ 0 & \cos\varphi_i \end{pmatrix}, \tag{5.110}$$

where

$$p_i = \frac{\sin\varphi_i}{\cos\varphi_i} = \tan\varphi_i, \quad \cos\varphi_i \neq 0 \quad \text{and} \quad u_i = \sin\varphi_i\cos\varphi_i.$$

On the other hand, the DLU and ULD factorizations of H_{φ_i} and its inverse, $H_{\varphi_i}^{-1}$, respectively, are defined as

$$H_{\varphi_i} = \begin{pmatrix} \cos\varphi_i & \sin\varphi_i \\ \sin\varphi_i & -\cos\varphi_i \end{pmatrix} = \begin{pmatrix} \cos\varphi_i & 0 \\ 0 & -\frac{1}{\cos\varphi_i} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -u_i & 1 \end{pmatrix} \begin{pmatrix} 1 & p_i \\ 0 & 1 \end{pmatrix},$$

$$H_{\varphi_i}^{-1} = \begin{pmatrix} \cos\varphi_i & \sin\varphi_i \\ \sin\varphi_i & -\cos\varphi_i \end{pmatrix} = \begin{pmatrix} 1 & -p_i \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ u_i & 1 \end{pmatrix} \begin{pmatrix} \frac{1}{\cos\varphi_i} & 0 \\ 0 & -\cos\varphi_i \end{pmatrix}, \quad (5.111)$$

where multipliers p_i and u_i are the same as in (5.110). The DLU and ULD structures of G_{φ_i} and its inverse, $G_{\varphi_i}^{-1}$, respectively, are shown in Fig. 5.12(a). The DLU and ULD structures of H_{φ_i} and its inverse, $H_{\varphi_i}^{-1}$, respectively, are shown in Fig. 5.12(b). Equations (5.106) and (5.110) imply that the Givens–Jacobi rotations G_{φ_i} can be represented by LUL or DLU factorization. To simplify the construction of multiplierless approximations of fast transforms, the concept of scaled transform can be applied and any plane rotation at the end of the signal flow graph can be replaced by the proper DLU structure involving only two multipliers. Elements of the diagonal matrix D are simply absorbed into the quantization stage [46].

Similarly, in practical implementations of the DLU structure of G_{φ_i} (or H_{φ_i}) there is a problem related to numerical instability for some rotation angles [46]. Specifically, the signal at the point V in DLU structure of G_{φ_i} shown in Fig. 5.12 (a) can be expressed as

$$V = u_i x_0 + (1 - p_i u_i) x_1 = \sin\varphi_i \cos\varphi_i x_0 + \cos^2\varphi_i x_1. \quad (5.112)$$

If the rotation angle φ_i is close to $k\pi + \frac{\pi}{2}$, where k is an integer, then the value of $1 - p_i u_i$, i.e., $\cos^2\varphi_i$ would be very small. For example, for the rotation angle $\frac{7\pi}{16}$, $\cos^2(\frac{7\pi}{16}) = 0.038060$. Therefore, a large relative error for $1 - p_i u_i$ could result when multipliers p_i and u_i are truncated or rounded, leading to a drastic change in frequency response of the result.

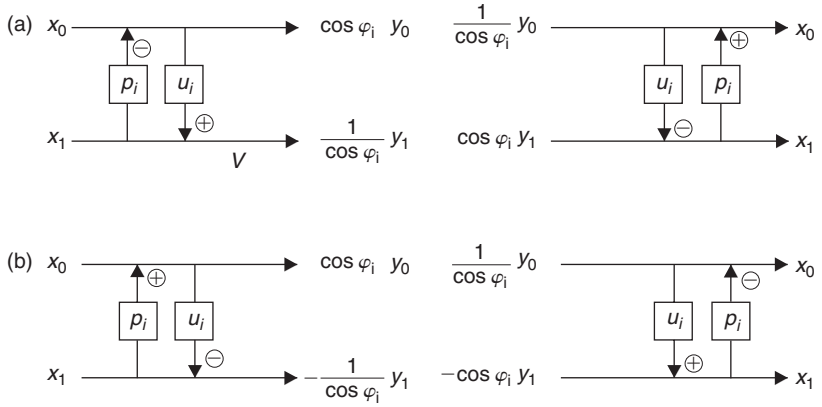


Fig. 5.12. The DLU and ULD structures: (a) of G_{φ_i} and its inverse $G_{\varphi_i}^{-1}$ and (b) of H_{φ_i} and its inverse $H_{\varphi_i}^{-1}$.

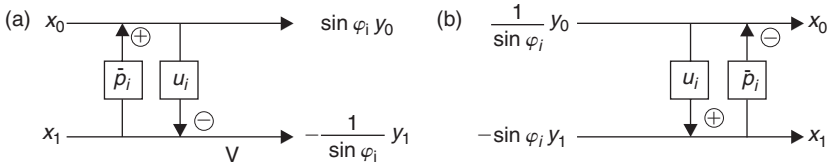


Fig. 5.13. The permuted DLU and ULD structures: (a) of G_{φ_i} and (b) its inverse $G_{\varphi_i}^{-1}$.

Another problem is that the multiplier $p_i = \tan\varphi_i$ would be much greater than 1. This fact increases the dynamic range of intermediate result and it is undesirable in both software and hardware implementations.

To eliminate the effect of finite-length approximations of multipliers on the performance of multiplierless approximations of fast transforms, an alternative equivalent permuted DLU factorization of G_{φ_i} is suggested and it is given by [46]

$$G_{\varphi_i} = \begin{pmatrix} \cos\varphi_i & -\sin\varphi_i \\ \sin\varphi_i & \cos\varphi_i \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \sin\varphi_i & 0 \\ 0 & -\frac{1}{\sin\varphi_i} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -u_i & 1 \end{pmatrix} \begin{pmatrix} 1 & \bar{p}_i \\ 0 & 1 \end{pmatrix}, \quad (5.113)$$

where

$$\bar{p}_i = \frac{1}{\tan\varphi_i} \quad \text{and} \quad u_i = \cos\varphi_i \sin\varphi_i.$$

The corresponding permuted DLU and ULD structures of G_{φ_i} and its inverse, $G_{\varphi_i}^{-1}$, are respectively shown in Fig. 5.13 (a) and (b). Evaluating the signal at the point V in the permuted DLU structure of G_{φ_i} shown in Fig. 5.13 (a) with the new multipliers $\bar{p}_i$ and u_i we get

$$V = -u_i x_0 + (1 - \bar{p}_i u_i) x_1 = -\sin\varphi_i \cos\varphi_i x_0 + \sin^2\varphi_i x_1, \quad (5.114)$$

and comparing with (5.112) the coefficient of x_1 changes from $\cos^2\varphi_i$ to $\sin^2\varphi_i$ which is more robust to truncation errors for rotation angles φ_i close to $k\pi + \frac{\pi}{2}$.

In general, when the DLU structure of G_{φ_i} is used to obtain finite-length approximation of fast transform with minimal dynamic range and balanced performance, its implementation can be controlled as follows. The original DLU structure shown in Fig. 5.12 (a) should be used if $\cos^2\varphi_i > \sin^2\varphi_i$, and the permuted DLU computational structure shown in Fig. 5.13 (a) should be adopted if $\cos^2\varphi_i < \sin^2\varphi_i$. When $\cos^2\varphi_i = \sin^2\varphi_i$ both structures reduce to the unnormalized Haar transform. Similar sensitivity analysis can be performed for the DLU structure of H_{φ_i}.

Whereas G_{φ_i} and H_{φ_i} rotation matrices are orthogonal (G_{φ_i} is eigenorthogonal while H_{φ_i} is non-eigenorthogonal), the factored Gauss–Jordan elementary matrices in LUL and DLU factorizations are not orthogonal. However, they are invertible. The invertibility guarantees perfect reconstruction between forward and inverse transformations.

If the multipliers p_i and u_i are floating-point numbers, then LUL structures shown in Fig. 5.11 represent the fast Givens–Jacobi rotation. If a *floor* operator (the largest integer that is less than or equal to the operand), a *ceil* operator (the smallest integer that is greater than or equal to the operand) or a *round* operator (see Section 5.5.2) is applied to each multiplier p_i and u_i, then the LUL and DLU structures can map integers to integers without destroying the perfect reconstruction property. If each step in the computational structure is chosen to be dyadic, then all multipliers can be replaced by finite-length binary numbers.

5.4.4.2 Dyadic rational numbers

The dyadic rational number is a binary fractional one of the form $\frac{k}{2^b}$, where $k, b \in N$ and k is an odd integer [50, 51]. Multiplication by dyadic rational number can be implemented using only binary arithmetic. The multiplicand is first multiplied by numerator k and the result is shifted to the right by b-bits. Neglecting shift operations, the minimum number of additions required for implementing a given binary fraction is equal to that for implementing its numerator k. An integer multiplication is equivalent to bit- shifting the multiplicand to the left by different numbers of bits and summing up these bit-shifted versions. The total number of shifts and additions required can be counted from the binary representation of the integer multiplier. For example, multiplication by $5 = (101)_2$ can be implemented by 1 addition and 1 shift. Similarly, multiplication by $7 = (111)_2$ can be implemented by 2 additions and 2 shifts, since $7 = 4 + 2 + 1 = (100)_2 + (10)_2 + (1)_2$. However, this is not the minimum number of additions needed to multiply a number by 7, because if we express $7 = 8 - 1 = (1000)_2 - (1)_2$, it is immediately clear that only 1 addition and 1 shift are required. This fact is related to minimum-adder representation of integer multiplier k which is based on the concept of multiplicative irreducibility in terms of adders [50, 51].

Definition 5.1: (Multiplicative irreducibility)
A positive integer multiplier x is said to be multiplicatively irreducible in terms of adders if the minimum number of adders required to implement its multiplication is equal to $n_x - 1$, where n_x is the number of 1s in the binary representation of x.

The multiplicative irreducibility is important in determining the minimum-adder representation of an integer multiplier.

Definition 5.2: (Minimum-adder representation)
An integer x can be decomposed into the form $x = a - b$, where $a, b \in \mathcal{N}$ are multiplicatively irreducible containing n_a and n_b binary 1s, respectively. The minimum number of adders required to implement the multiplication by x is $n_a + n_b - 1$.

Definition 5.3: (Irreducible form)
Given an integer x, minimum-adder representation is said to be the irreducible form of x.

An algorithm for systematically finding the multiplierless approximations of fast transforms with minimum-adder representation while minimizing MSE approximation error or maximizing the performance of approximated transform is presented in Refs. [50, 51].

Note: The so-called SOPOT (Sum-Of-Powers-Of-Two) approximation of a floating-point multiplier was introduced in Refs. [52, 53]. The SOPOT approximation is defined as

$$\sum_{k=1}^{M} a_k\, 2^{b_k}, \quad a_k \in \{-1, 0, 1\},\ b_k \in \{-r, \ldots, -1, 0, 1, \ldots, r\},$$

where r is the range of multiplier and M is the number of terms. Thus, each floating-point multiplier can be replaced by the limited number of additions and shifts. However, the SOPOT approximation is actually a dyadic rational number after summing up the terms.

5.4.4.3 Dyadic multiplierless approximations of $G_{\frac{3\pi}{8}}$ and $H_{\frac{\pi}{4}}$

In the following we present in detail the examples of how to replace:

- Givens–Jacobi rotation G_φ and its inverse, $G_{-\varphi}$, by LUL structures with dyadic approximations of multipliers in terms of minimum-adder representation for rotation angle $\varphi = \frac{3\pi}{8}$.
- Householder reflection H_φ and its inverse, H_φ^{-1}, by DLU and ULD structures, respectively, with dyadic approximations of multipliers in terms of minimum-adder representation for rotation angle $\varphi = \frac{\pi}{4}$.

Consider the LUL factorizations of G_φ and its inverse, $G_{-\varphi}$, given by (5.106) and their corresponding LUL structures shown in Fig. 5.11 (a) and (b), respectively. For rotation angle $\varphi = \frac{3\pi}{8}$ the values of multipliers p and u may have the following dyadic approximations:

p	0.668179	u	0.923880
3/4	0.750000	1	1.000000
5/8	0.625000	7/8	0.875000
11/16	0.687500	15/16	0.937500
21/32	0.656250	29/32	0.906250

As a tradeoff between computational cost and approximation precision we choose the dyadic approximations 5/8 and 15/16 for multipliers p and u, respectively. Substituting the dyadic approximations of p and u into the LUL structure we get the dyadic form of $G_{\frac{3\pi}{8}}$ denoted by $\bar{G}_{\frac{3\pi}{8}}$, (5/8,15/16). The scheme for replacing $G_{\frac{3\pi}{8}}$ by its dyadic form $\bar{G}_{\frac{3\pi}{8}}$ (5/8,15/16) is shown in Fig. 5.14 (a), and for replacing $G_{-\frac{3\pi}{8}}$ by its dyadic form $\bar{G}_{-\frac{3\pi}{8}}$ (5/8,15/16) is shown in Fig. 5.14 (b).

Since the dyadic multipliers can be decomposed into minimum-adder representation as

$$\frac{5}{8} = \frac{4+1}{8} = \frac{1}{2} + \frac{1}{8} \quad \text{requiring 1 addition and 2 shifts,}$$

$$\frac{15}{16} = \frac{16-1}{16} = 1 - \frac{1}{16} \quad \text{requiring 1 addition and 1 shift,}$$

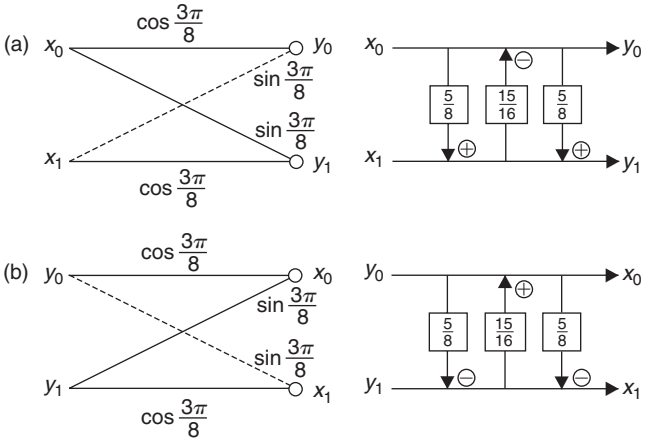


Fig. 5.14. Replacing: (a) $G_{\frac{3\pi}{8}}$ by its dyadic form $\bar{G}_{\frac{3\pi}{8}}(5/8, 15/16)$ and (b) $G_{-\frac{3\pi}{8}}$ by its dyadic form $\bar{G}_{-\frac{3\pi}{8}}$ (5/8,15/16).

and taking into account 3 additions required by the LUL structure, the total computational cost in implementing the dyadic form $\bar{G}_{\frac{3\pi}{8}}$ (5/8,15/16) as well as $\bar{G}_{-\frac{3\pi}{8}}$ (5/8,15/16) is 6 additions and 5 shifts.

Now consider the DLU and ULD factorizations of H_φ and its inverse, H_φ^{-1}, respectively, given by (5.111) and their corresponding DLU and ULD structures shown in Fig. 5.12 (b). For rotation angle $\varphi = \frac{\pi}{4}$ the values of multipliers are $p = 1$ and $u = \frac{1}{2}$, which are already in dyadic form. By direct substitution of $p = 1$ and $u = \frac{1}{2}$ into DLU factorization (5.111) we get

$$H_{\frac{\pi}{4}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & -\sqrt{2} \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{1}{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 \\ 0 & \sqrt{2} \end{pmatrix} \begin{pmatrix} 1 & 1 \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}, \tag{5.115}$$

and $H_{\frac{\pi}{4}}$ is reduced to the unnormalized Haar transform. Denote the dyadic form of $H_{\frac{\pi}{4}}$ as $D\bar{H}_{\frac{\pi}{4}}$ (1,1/2), where $D = \text{diag}\{\sqrt{2}/2, -\sqrt{2}\}$. The scheme for replacing $H_{\frac{\pi}{4}}$ by its dyadic form $D\bar{H}_{\frac{\pi}{4}}(1, 1/2)$ is shown in Fig. 5.15 (a), and for replacing $H_{\frac{\pi}{4}}^{-1}$ by its dyadic form $\bar{H}_{\frac{\pi}{4}}^{-1}(1, 1/2)D^{-1}$ is shown in Fig. 5.15 (b).

Provided the diagonal elements of D are incorporated into the quantization stage, the total computational cost in implementing the dyadic form $D\bar{H}_{\frac{\pi}{4}}(1, 1/2)$ as well as $\bar{H}_{\frac{\pi}{4}}^{-1}(1, 1/2)D^{-1}$ is 2 additions and 1 shift.

5.4.4.4 Construction of BinDCT/BinDST and IntDCT/IntDST

In the following is described the construction and implementation of multiplierless approximations of fast DCTs and DSTs. In order to uniquely distinguish the multiplierless

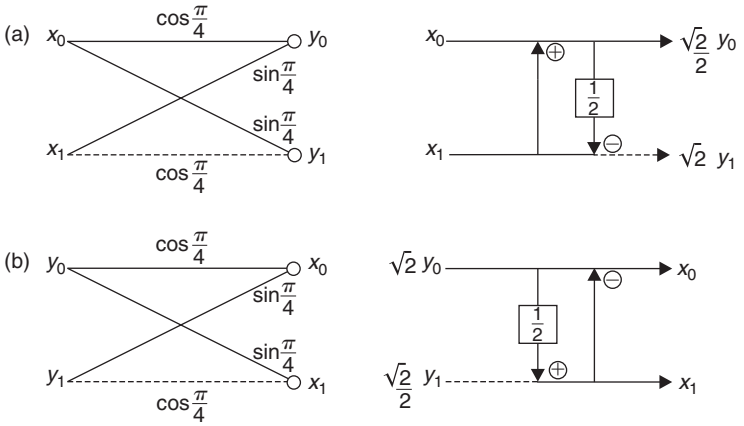


Fig. 5.15. Replacing: (a) $H_{\frac{\pi}{4}}$ by its dyadic form $D\bar{H}_{\frac{\pi}{4}}(1, 1/2)$ and (b) $H^{-1}_{\frac{\pi}{4}}$ by its dyadic form $\bar{H}^{-1}_{\frac{\pi}{4}}(1, 1/2)D^{-1}$.

approximation of a specific DCT and DST, we will use the notation BinDCT-*x*/BinDST-*x* and IntDCT-*x*/IntDST-*x*, where *x* denotes the type I, II, III or IV.

Generally, the construction of multiplierless approximation of a given fast DCT/DST consists of two common steps:

1. To consider a plane rotation-based sparse matrix factorization of DCT/DST matrix defining its fast algorithm which is equivalently represented by a corresponding generalized signal flow graph.
2. To convert each plane rotation to G_{φ_i} or H_{φ_i}, if it is necessary, and then to factorize it into LUL or DLU form. Finally, to replace the G_{φ_i} or H_{φ_i} by the corresponding LUL or DLU form with dyadic approximation of multipliers.

In almost all cases for a given N, it is sufficient to consider the signal flow graph. After the approximation of all plane rotations we obtain immediately the forward and inverse fast multiplierless BinDCT and IntDCT implementations. The construction of BinDCT and IntDCT is illustrated for 8-point DCT-II and its inverse, DCT-III. Constructing BinDCT-II and IntDCT-II and their inverses, BinDCT-III and IntDCT-III, are based on the following fast algorithms:

- Fast algorithm for the DCT-II computation and its inverse, DCT-III [17] (see Section 4.4.3.7). Their fast multiplierless approximations are denoted as BinDCT-IIC and BinDCT-IIIC. The construction of BinDCT-IIC and BinDCT-IIIC is presented as complete as possible.
- Fast algorithm for the scaled DCT-II and DCT-III computation [20] (see Section 4.4.3.1). Their multiplierless approximations are denoted as BinDCT-IIL and BinDCT-IIIL.

- Fast split-radix algorithm for the DCT-II and DCT-III computation (see Section 4.4.3.4). Their multiplierless approximations are denoted as BinDCT-IIS and BinDCT-IIIS.
- The fast algorithm DCT-II and DCT-III computation via WHT [18, 19] (see Section 4.4.3.2). Their multiplierless approximations are denoted as IntDCT-II and IntDCT-III.

5.4.4.5 Construction of BinDCT-IIC and BinDCT-IIIC

Let $\hat{C}_8^{\mathrm{II}}$ be the DCT-II matrix with its rows in bit-reversed order. Then, according to the recursive sparse matrix factorization (4.50) it can be written as

$$\hat{C}_8^{\mathrm{II}} = \hat{D}_8 \begin{pmatrix} \begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & C_2^{\mathrm{IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \hat{C}_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.116}$$

where

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} \end{pmatrix} = H_{\frac{\pi}{4}},$$

$$C_2^{\mathrm{IV}} J_2 = \begin{pmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix} = G_{-\frac{3\pi}{8}}, \tag{5.117}$$

and among the sparse matrix factorizations of $\hat{C}_4^{\mathrm{IV}} J_4$ (see Section 5.4.3) the most suitable form is given by

$$\begin{aligned} \hat{C}_4^{\mathrm{IV}} J_4 &= \begin{pmatrix} \cos\frac{7\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{7\pi}{16} \\ \cos\frac{3\pi}{16} & \cos\frac{7\pi}{16} & -\sin\frac{7\pi}{16} & \sin\frac{3\pi}{16} \\ -\sin\frac{3\pi}{16} & -\sin\frac{7\pi}{16} & -\cos\frac{7\pi}{16} & \cos\frac{3\pi}{16} \\ -\sin\frac{7\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{7\pi}{16} \end{pmatrix} \\ &= \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{4} & \sin\frac{\pi}{4} & 0 \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}. \end{aligned} \tag{5.118}$$

The diagonal matrix $\hat{D}_8$ in (5.116) for the normalization of DCT-II coefficients is given by $\hat{D}_8 = \mathrm{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. The generalized signal flow graph for the fast 8-point

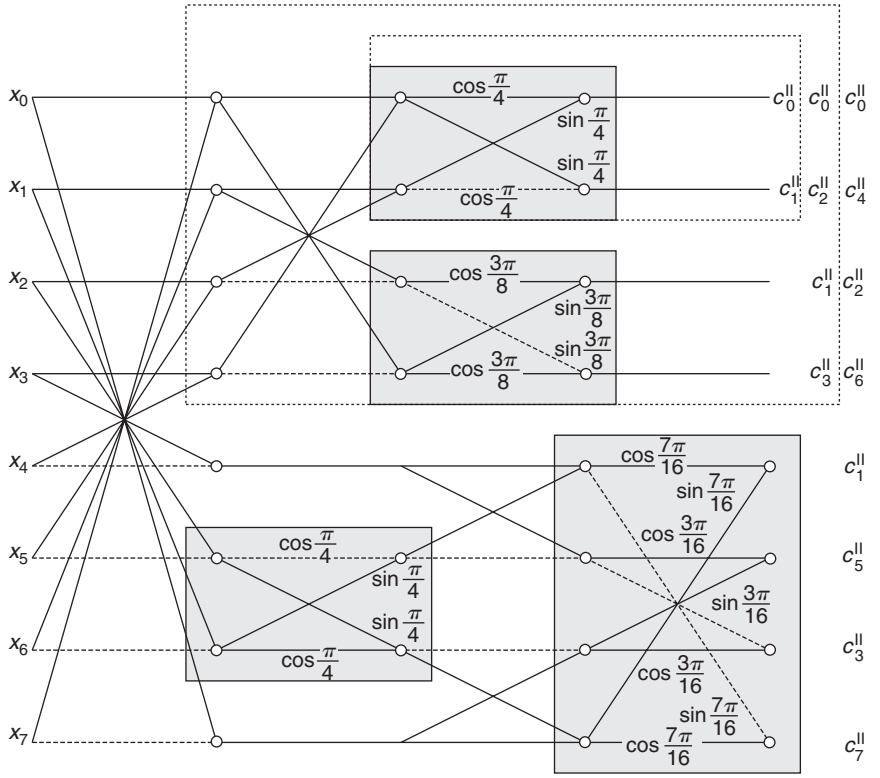


Fig. 5.16. The generalized signal flow graph for the fast 2-, 4- and 8-point DCT-II computation and its inverse, DCT-III, based on (5.116)–(5.118) with highlighted plane rotations.

DCT-II computation with embedded 2- and 4-point transforms based on (5.116)–(5.118) with highlighted plane rotations is shown in Fig. 5.16. The computation of the 8-point DCT-II requires 13 floating-point multiplications and 29 floating-point additions. Comparing Figs. 4.8 with 5.16 we note that two butterflies with multipliers $\cos\frac{\pi}{4}$ in Fig. 4.8 have been replaced by plane rotations with the angle $\frac{\pi}{4}$ in Fig. 5.16.

The signal flow graph in Fig. 5.16 involves 5 plane rotations, specifically,

$$H_{\frac{\pi}{4}}, \quad G_{-\frac{3\pi}{8}}, \quad \bar{R}_{\frac{\pi}{4}} = G_{-\frac{\pi}{4}} \begin{pmatrix} -1 & 1 \\ 0 & 1 \end{pmatrix}, \quad G_{-\frac{3\pi}{16}} \text{ and } G_{-\frac{7\pi}{16}}.$$

Now all that is needed is to replace each plane rotation by the proper LUL or DLU structure with dyadic approximation of multipliers p_i and u_i, $i = 0, 1, 2, 3, 4$. Replacing $G_{-\frac{3\pi}{8}}$ by the LUL structure with dyadic approximation of multipliers is shown in Fig. 5.14 (b), while replacing $H_{\frac{\pi}{4}}$ by the DLU structure is shown in Fig. 5.15 (a). The remaining plane rotations $\bar{R}_{\frac{\pi}{4}}$, $G_{-\frac{3\pi}{16}}$ and $G_{-\frac{7\pi}{16}}$ in the lower half of the signal flow graph in Fig. 5.16 are similarly replaced by LUL structures. In Table 5.17 are summarized all plane rotations with analytical values of the multipliers p_i, u_i, $i = 0, 1, 2, 3, 4$ together with their dyadic

Table 5.17. Plane rotations with analytical values of the multipliers, their dyadic approximations and the computational cost of DLU and LUL structures in BinDCT-IIC.

Plane rotation	Multiplier	Analytical value	Dyadic approximation	Computational structure	Computational cost (adds + shifts)
$H_{\frac{\pi}{4}}$	p_0		1	DLU	
	u_0		1/2	$D\bar{H}_{\frac{\pi}{4}}\ (1, 1/2)$	2A + 1S
$G_{-\frac{3\pi}{8}}$	p_1	0.668179	$5/8 \approx 0.625000$	LUL	
	u_1	0.923880	$15/16 \approx 0.937500$	$\bar{G}_{-\frac{3\pi}{8}}\ (5/8, 15/16)$	6A + 5S
				LUL	
$\bar{R}_{\frac{\pi}{4}}$	p_2	0.414214	$7/16 \approx 0.437500$	$\bar{G}_{-\frac{\pi}{4}}\ (7/16, 3/4) \times$	
	u_2	0.707107	$3/4 \approx 0.750000$	$\text{diag}\{-1, 1\}$	6A + 6S
$G_{-\frac{3\pi}{16}}$	p_3	0.303347	$1/4 \approx 0.250000$	LUL	
	u_3	0.555570	$1/2 \approx 0.500000$	$\bar{G}_{-\frac{3\pi}{16}}\ (1/4, 1/2)$	3A + 3S
$G_{-\frac{7\pi}{16}}$	p_4	0.820679	$13/16 \approx 0.812500$	LUL	
	u_4	0.980785	1	$\bar{G}_{-\frac{7\pi}{16}}\ (13/16, 1)$	7A + 6S

approximations and the computational cost of DLU and LUL structures in terms of add and shift operations.

The general forms of the forward fast multiplierless 8-point BinDCT-IIC and its inverse, BinDCT-IIIC, with embedded 2- and 4-point transforms are shown in Fig. 5.17 (a) and (b), respectively. The diagonal matrix $\hat{D}_8$ for the normalization of coefficients of fast multiplierless BinDCT-IIC is given by $\hat{D}_8 = \text{diag}\{\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. The fast multiplierless BinDCT-IIIC implementation in Fig. 5.17 (b) is obtained by reversing the BinDCT-IIC signal flow graph and taking inverses of the DLU and LUL structures. If multipliers p_i and u_i are floating-point numbers, then implementations of the 8-point DCT-II in Figs. 5.16 and 5.17 (a) are equivalent. Any other dyadic approximations of p_i and u_i can be substituted into the corresponding DLU and LUL structures in Table 5.17 to obtain various configurations of BinDCT-IIC with different accuracies and computational costs. We note that plane rotations $G_{-\frac{3\pi}{8}}$, $G_{-\frac{3\pi}{16}}$ and $G_{-\frac{7\pi}{16}}$ at the end of the signal flow graph in Fig. 5.16 can be alternatively replaced by DLU structures. Such BinDCT-IIC configurations can be found in Refs. [46, 50, 51].

For the dyadic approximations of multipliers p_i and u_i shown in Table 5.17, if we use them with the appropriate DLU and LUL forms of H_{φ_i} and G_{φ_i} and multiply the factored Gauss–Jordan elementary matrices followed by substitution of the resulting matrices into (5.117) and (5.118), we obtain the lower-order approximated matrices $\bar{C}_2^{\text{II}}$, $\bar{C}_2^{\text{IV}} J_2$ and $\hat{\bar{C}}_4^{\text{IV}} J_4$ as

$$\bar{C}_2^{\text{II}} = \begin{pmatrix} 1 & 1 \\ \frac{1}{2} & -\frac{1}{2} \end{pmatrix}, \quad \bar{C}_2^{\text{IV}} J_2 = \begin{pmatrix} \frac{53}{128} & \frac{15}{16} \\ -\frac{905}{1024} & \frac{53}{128} \end{pmatrix},$$

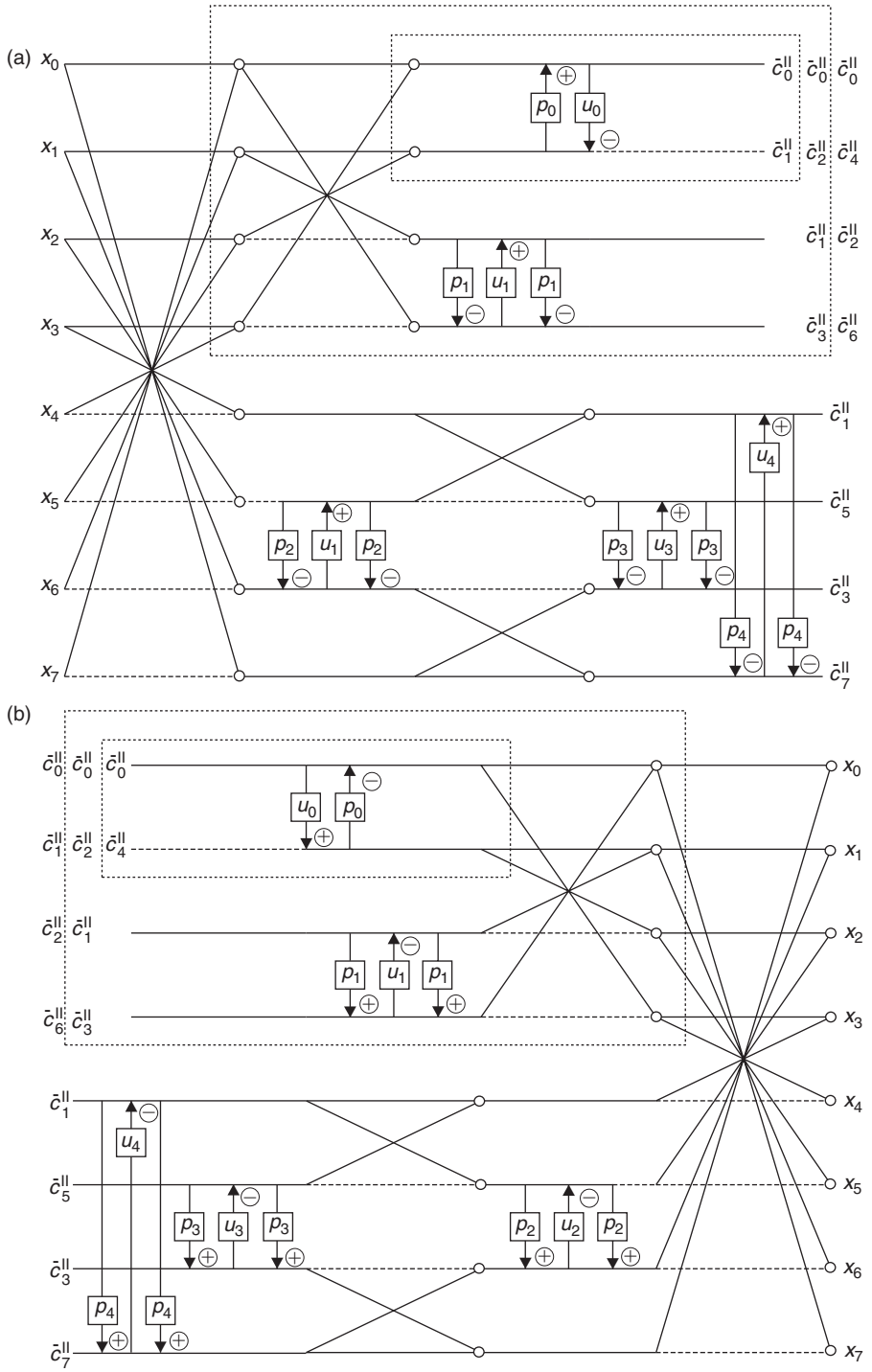


Fig. 5.17. The general forms of the forward fast multiplierless 8-point (a) BinDCT-IIC, (b) and its inverse, BinDCT-IIIC, with embedded 2- and 4-point transforms.

$$\hat{\tilde{C}}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \frac{3}{16} & \frac{155}{256} & \frac{13}{16} & 1 \\ \frac{7}{8} & \frac{455}{2048} & -\frac{127}{128} & \frac{1}{2} \\ -\frac{15}{32} & -\frac{7823}{8192} & -\frac{121}{512} & \frac{7}{8} \\ -\frac{247}{256} & \frac{3217}{4096} & -\frac{153}{256} & \frac{3}{16} \end{pmatrix}.$$

Substituting the above lower-order approximated matrices into the sparse factorization (5.116) the BinDCT-IIC basis vectors approximating the DCT-II matrix $\hat{C}_8^{\mathrm{II}}$ are given by

$$\hat{\tilde{C}}_8^{\mathrm{II}} = \hat{D}_8 \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} & \frac{1}{2} & -\frac{1}{2} & -\frac{1}{2} & \frac{1}{2} \\ \frac{15}{16} & \frac{53}{128} & -\frac{53}{128} & -\frac{15}{16} & -\frac{15}{16} & -\frac{53}{128} & \frac{53}{128} & \frac{15}{16} \\ \frac{53}{128} & -\frac{905}{1024} & \frac{905}{1024} & -\frac{53}{128} & -\frac{53}{128} & \frac{905}{1024} & -\frac{905}{1024} & \frac{53}{128} \\ 1 & \frac{13}{16} & \frac{155}{256} & \frac{3}{16} & -\frac{3}{16} & -\frac{155}{256} & -\frac{13}{16} & -1 \\ \frac{1}{2} & -\frac{127}{128} & \frac{455}{2048} & \frac{7}{8} & -\frac{7}{8} & -\frac{455}{2048} & \frac{127}{128} & -\frac{1}{2} \\ \frac{7}{8} & -\frac{121}{512} & -\frac{7823}{8192} & -\frac{15}{32} & \frac{15}{32} & \frac{7823}{8192} & \frac{121}{512} & -\frac{7}{8} \\ \frac{3}{16} & -\frac{153}{256} & \frac{3217}{4096} & -\frac{247}{256} & \frac{247}{256} & -\frac{3217}{4096} & \frac{153}{256} & -\frac{3}{16} \end{pmatrix},$$

where $\hat{D}_8 = \mathrm{diag}\{\frac{\sqrt{2}}{4}, \frac{\sqrt{2}}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$.

In Table 5.21 is summarized the MSE approximation error, performance measures of the fast multiplierless BinDCT-IIC compared with the 8-point DCT-II, and the total computational complexity of BinDCT-IIC and BinDCT-IIIC for specific dyadic approximations of the multipliers from Table 5.17.

5.4.4.6 Construction of BinDCT-IIL and BinDCT-IIIL

The fast algorithm for the 8-point scaled DCT-II and its inverse, DCT-III [20], with respect to (5.116) and (5.117) is based on the sparse factorization of the scaled transform matrix $\hat{C}_8^{\mathrm{II}}$ defined as

$$\hat{C}_8^{\mathrm{II}} = \hat{D}_8 \begin{pmatrix} \begin{pmatrix} \sqrt{2}C_2^{\mathrm{II}} & 0 \\ 0 & \sqrt{2}J_2 C_2^{\mathrm{IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & \sqrt{2}J_4 \hat{C}_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.119}$$

where

$$\sqrt{2}C_2^{\mathrm{II}} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix},$$

$$\sqrt{2}C_2^{\mathrm{IV}} J_2 = \sqrt{2} \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix} = \sqrt{2} G_{-\frac{3\pi}{8}}, \tag{5.120}$$

and the sparse factorization of $\sqrt{2}J_4 C_4^{\mathrm{IV}} J_4$ is given by

$$\begin{aligned}
\sqrt{2}J_4\hat{C}_4^{\mathrm{IV}}J_4 &= \sqrt{2}\begin{pmatrix} -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ -\sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} \end{pmatrix} \\
&= \sqrt{2}\begin{pmatrix} -\cos\frac{\pi}{4} & 0 & 0 & \sin\frac{\pi}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\frac{\pi}{4} & 0 & 0 & \cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\frac{3\pi}{16} & 0 & 0 & \sin\frac{3\pi}{16} \\ 0 & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & 0 \\ 0 & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & 0 \\ -\sin\frac{3\pi}{16} & 0 & 0 & \cos\frac{3\pi}{16} \end{pmatrix} \\
&= \begin{pmatrix} -1 & 0 & 0 & 1 \\ 0 & \sqrt{2} & 0 & 0 \\ 0 & 0 & \sqrt{2} & 0 \\ 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & -1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\frac{3\pi}{16} & 0 & 0 & \sin\frac{3\pi}{16} \\ 0 & \cos\frac{\pi}{16} & \sin\frac{\pi}{16} & 0 \\ 0 & -\sin\frac{\pi}{16} & \cos\frac{\pi}{16} & 0 \\ -\sin\frac{3\pi}{16} & 0 & 0 & \cos\frac{3\pi}{16} \end{pmatrix}.
\end{aligned} \tag{5.121}$$

The diagonal matrix $\hat{D}_8$ in (5.119) is given by $\hat{D}_8 = \mathrm{diag}\{\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}\}$. The generalized signal flow graph for the fast 8-point scaled DCT-II with embedded 2- and 4-point transforms based on (5.119)–(5.121) with highlighted plane rotations is shown in Fig. 5.18. The computation of 8-point scaled DCT-II requires 11 floating-point multiplications and 29 floating-point additions. In the construction of BinDCT-IIL the scaling factors $\sqrt{2}$ in (5.120) and (5.121) are incorporated into the diagonal matrix having the form $\hat{D}_8 = \mathrm{diag}\{\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{8}}, \frac{1}{2}, \frac{1}{2}, \frac{1}{\sqrt{8}}\}$.

The signal flow graph in Fig. 5.18 involves 3 plane rotations, specifically,

$$G_{-\frac{3\pi}{8}}, \quad G_{-\frac{\pi}{16}} \quad \text{and} \quad G_{-\frac{3\pi}{16}}.$$

In Table 5.18 are summarized all plane rotations with analytical values of the multipliers p_i, u_i, $i = 0, 1, 2$ together with their dyadic approximations and the computational cost of the corresponding LUL structures in terms of add and shift operations.

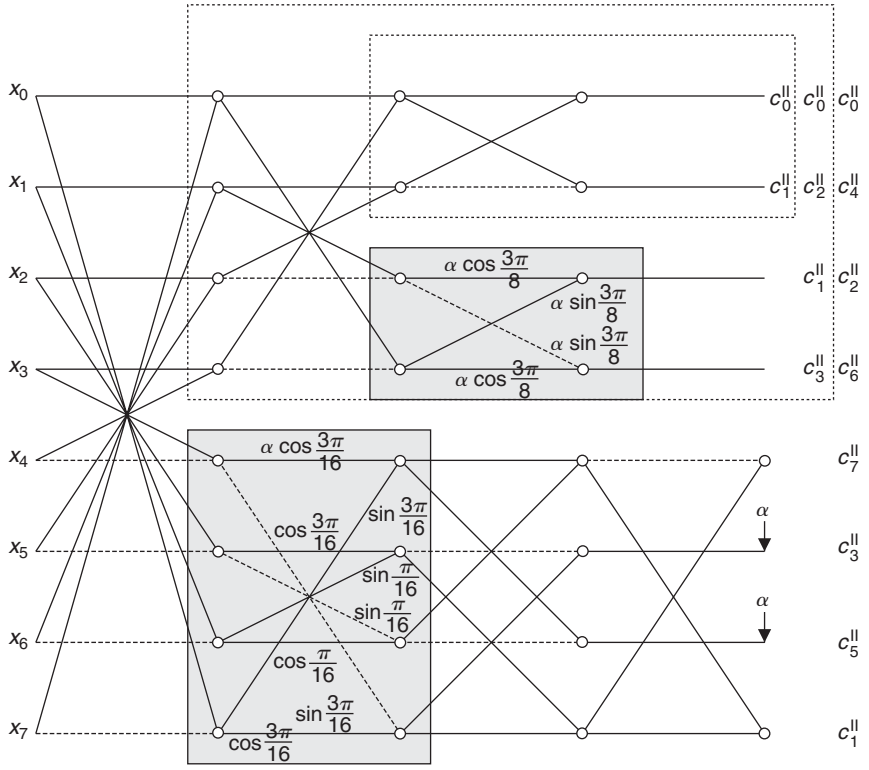


Fig. 5.18. The generalized signal flow graph for the fast 2-, 4- and 8-point scaled DCT-II computation and its inverse, DCT-III, based on (5.119)–(5.121) with highlighted plane rotations.

Table 5.18. Plane rotations with analytical values of the multipliers, their dyadic approximations and the computational cost of the corresponding LUL structures in BinDCT-IIL.

Plane rotation	Multiplier	Analytical value	Dyadic approximation	Computational structure	Computational cost (adds + shifts)
$G_{-\frac{3\pi}{8}}$	p_0	0.668179	$5/8 \approx 0.625000$	LUL	
	u_0	0.923880	$15/16 \approx 0.937500$	$\bar{G}_{-\frac{3\pi}{8}}$ (5/8, 15/16)	6A + 5S
$G_{-\frac{\pi}{16}}$	p_1	0.098491	$3/32 \approx 0.093750$	LUL	
	u_1	0.195090	$3/16 \approx 0.187500$	$\bar{G}_{-\frac{\pi}{16}}$ (3/32, 3/16)	7A + 8S
$G_{-\frac{3\pi}{16}}$	p_2	0.303347	$5/16 \approx 0.312500$	LUL	
	u_2	0.555570	$9/16 \approx 0.562500$	$\bar{G}_{-\frac{3\pi}{16}}$ (5/16, 9/16)	7A + 8S

The general form of the fast multiplierless 8-point BinDCT-IIL with embedded 2- and 4-point transforms is shown in Fig. 5.19. The fast multiplierless BinDCT-IIIL implementation is obtained by reversing the BinDCT-IIL signal flow graph and taking inverses of the LUL structures. Again, if multipliers p_i and u_i are floating-point numbers, then

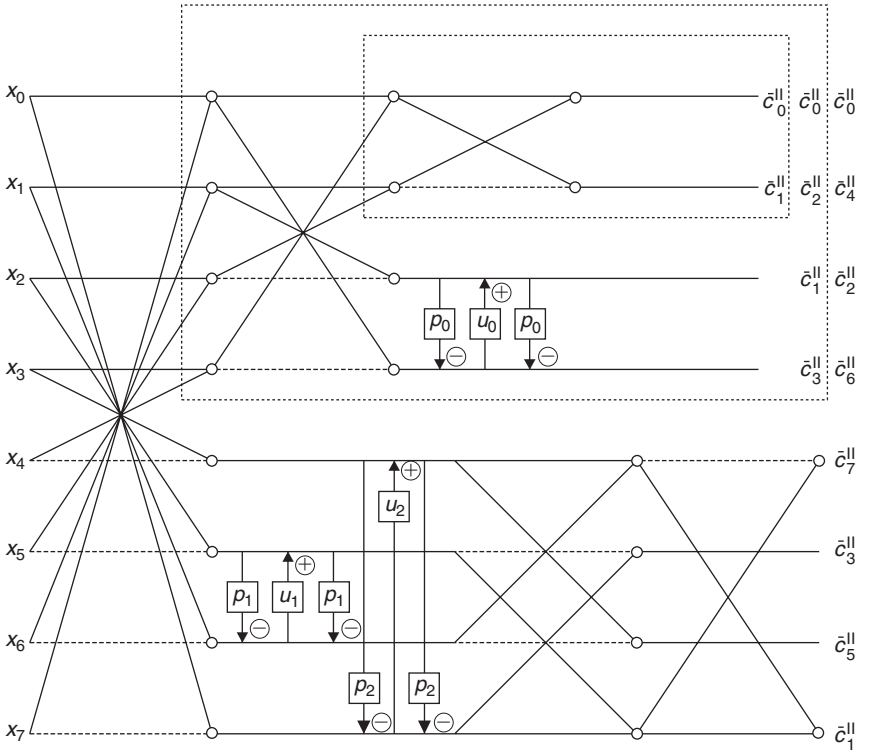


Fig. 5.19. The general form of the fast multiplierless 8-point BinDCT-IIL with embedded 2- and 4-point transforms.

implementations of the 8-point DCT-II in Figs. 5.18 and 5.19 are equivalent. Various other configurations of BinDCT-IIL can be found in Ref. [46].

In Table 5.21 is summarized the MSE approximation error, performance measures of the fast multiplierless BinDCT-IIL compared with the 8-point DCT-II, and the total computational complexity of BinDCT-IIL and BinDCT-IIIL for specific dyadic approximations of the multipliers from Table 5.18.

5.4.4.7 Construction of BinDCT-IIS and BinDCT-IIIS

Let $\hat{C}_8^{\mathrm{II}}$ be the DCT-II matrix with its rows in bit-reversed order. Then, according to the split-radix factorization given by (4.81) and (4.82) it can be written as

$$\hat{C}_8^{\mathrm{II}} = \hat{D}_8 \begin{pmatrix} \begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & K_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & K_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.122}$$

where

$$K_2 = \begin{pmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix} = G_{-\frac{3\pi}{8}}, \tag{5.123}$$

and

$$\begin{aligned} K_4 &= \begin{pmatrix} \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{3\pi}{16} \\ -\sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \\ 0 & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ 0 & -\cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ -\cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \end{pmatrix} \\ &= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & \cos\frac{5\pi}{16} & \sin\frac{5\pi}{16} & 0 \\ 0 & -\sin\frac{5\pi}{16} & \cos\frac{5\pi}{16} & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix}. \end{aligned} \tag{5.124}$$

The diagonal matrix $\hat{D}_8$ in (5.122) is given by $\hat{D}_8 = \text{diag}\,\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. It is easily seen that $K_2 = C_2^{\text{IV}} J_2$ and $K_4 = \hat{C}_4^{\text{IV}} J_4$. The generalized signal flow graph for the 8-point DCT-II computation with embedded 2- and 4-point transforms based on (5.122)–(5.124) with highlighted plane rotations is shown in Fig. 5.20. The computation of the 8-point DCT-II requires 13 floating-point multiplications and 29 floating-point additions. Comparing Figs. 4.14 with 5.20 we note that two butterflies with multipliers $\cos\frac{\pi}{4}$ in Fig. 4.14 have been replaced by plane rotations with the angle $\frac{\pi}{4}$ in Fig. 5.20. The signal flow graph in Fig. 5.20 involves 5 plane rotations, specifically,

$$H_{\frac{\pi}{4}}, \quad G_{-\frac{3\pi}{8}}, \quad G_{-\frac{5\pi}{16}}, \quad G_{-\frac{7\pi}{16}} \quad \text{and} \quad G_{\frac{\pi}{4}}.$$

In Table 5.19 we summarize all plane rotations with analytical values of the multipliers p_i, u_i, $i = 0, 1, 2, 3, 4$ together with their dyadic approximations and the computational cost of the DLU and LUL structures in terms of add and shift operations.

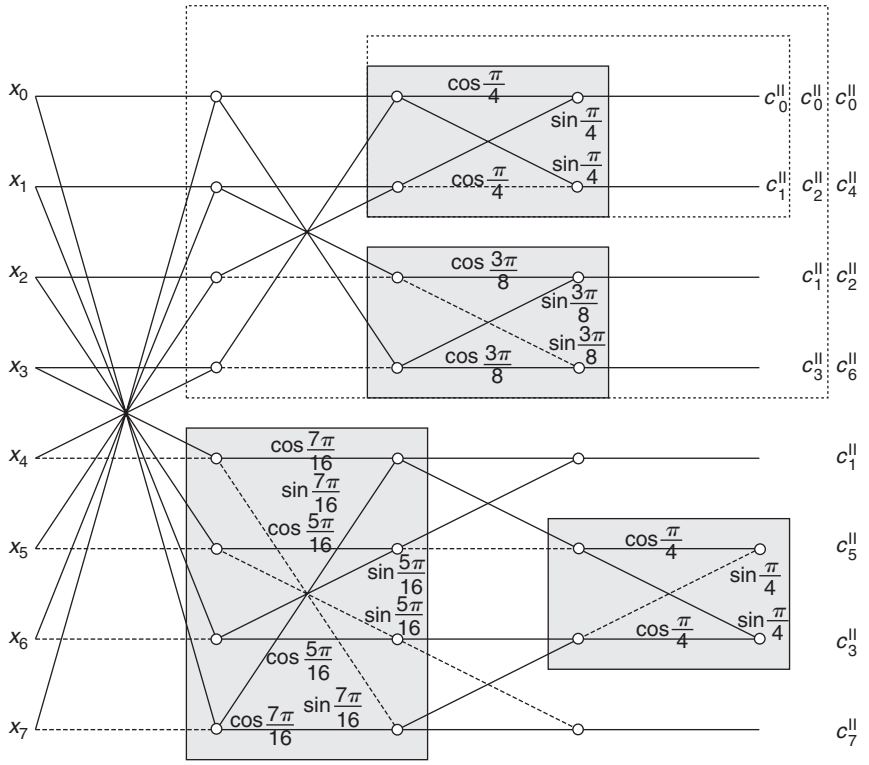


Fig. 5.20. The generalized signal flow graph for the fast 2-, 4- and 8-point DCT-II computation and its inverse, DCT-III, based on (5.122)–(5.124) with highlighted plane rotations.

Table 5.19. Plane rotations with analytical values of the multipliers, their dyadic approximations and the computational cost of DLU and LUL structures in BinDCT-IIS.

Plane rotation	Multiplier	Analytical value	Dyadic approximation	Computational structure	Computational cost (adds + shifts)
$H_{\frac{\pi}{4}}$	p_0		1	DLU	
	u_0		1/2	$D\tilde{H}_{\frac{\pi}{4}}(1, 1/2)$	2A + 1S
$G_{-\frac{3\pi}{8}}$	p_1	0.668179	$5/8 \approx 0.625000$	LUL	
	u_1	0.923880	$15/16 \approx 0.937500$	$\tilde{G}_{-\frac{3\pi}{8}}(5/8, 15/16)$	6A + 5S
$G_{-\frac{5\pi}{16}}$	p_2	0.534511	$17/32 \approx 0.531250$	LUL	
	u_2	0.831470	$13/16 \approx 0.812500$	$\tilde{G}_{-\frac{5\pi}{16}}(17/32, 13/16)$	7A + 6S
$G_{-\frac{7\pi}{16}}$	p_3	0.820679	$13/16 \approx 0.812500$	LUL	
	u_3	0.980785	$31/32 \approx 0.968750$	$\tilde{G}_{-\frac{7\pi}{16}}(13/16, 31/32)$	8A + 5S
$G_{\frac{\pi}{4}}$	p_4		1	DLU	
	u_4		1/2	$D\tilde{G}_{\frac{\pi}{4}}(1, 1/2)$	2A + 1S

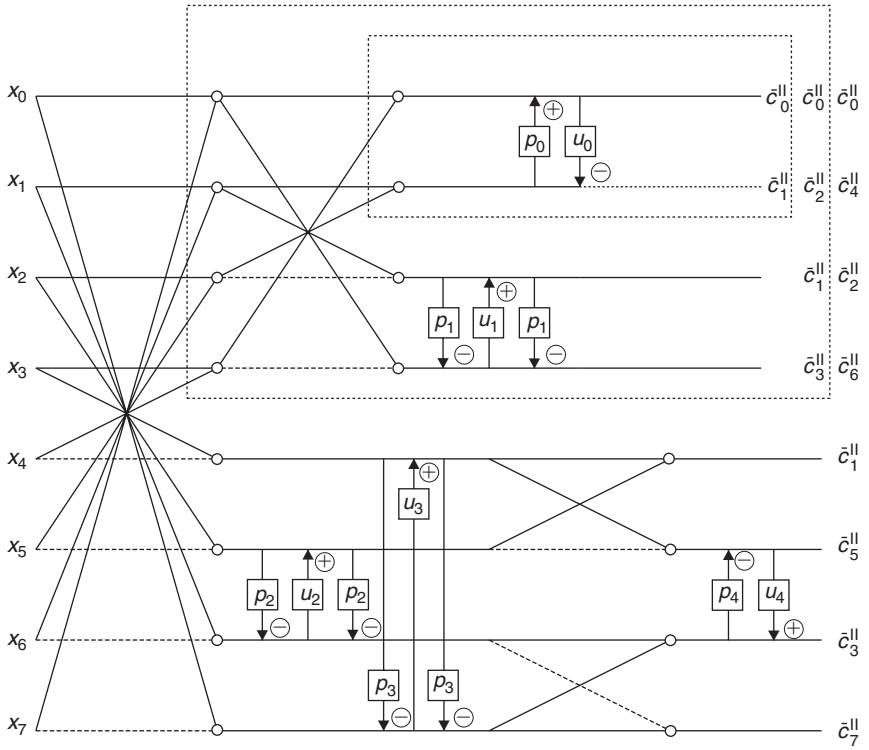


Fig. 5.21. The general form of the fast multiplierless 8-point BinDCT-IIS with embedded 2- and 4-point transforms.

The general form of the fast multiplierless 8-point BinDCT-IIS with embedded 2- and 4-point transforms is shown in Fig. 5.21. The fast BinDCT-IIIS implementation is obtained by reversing the BinDCT-IIS signal flow graph and taking inverses of the DLU and LUL structures. Again, if multipliers p_i and u_i are floating-point numbers, then implementations of the 8-point DCT-II in Figs. 5.20 and 5.21 are equivalent.

In Table 5.21 is summarized the MSE approximation error, performance measures of the fast multiplierless BinDCT-IIS compared with the 8-point DCT-II, and the total computational complexity of BinDCT-IIS and BinDCT-IIIS for specific dyadic approximations of the multipliers from Table 5.19.

5.4.4.8 Construction of IntDCT-II and IntDCT-III

The construction of fast multiplierless IntDCT-II is based on the DCT-II computation via WHT [18, 19]. According to (4.60) the DCT-II matrix $\hat{C}^{\text{II}}_8$ with its rows in bit-reversed order is defined as

$$\hat{C}^{\text{II}}_8 = \hat{D}_8 \, T_8 \, \hat{W}_8, \tag{5.125}$$

where $\hat{W}_8$ is the WHT matrix and T_8 is the conversion matrix. Block matrices U_2 and U_4 in the conversion matrix are given by

$$U_2 = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} = G_{-\frac{\pi}{8}}, \tag{5.126}$$

and

$$U_4 = P_4 \begin{pmatrix} \cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & 0 & 0 \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} & 0 & 0 \\ 0 & 0 & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ 0 & 0 & -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} P_4, \tag{5.127}$$

where P_4 is a permutation matrix. The diagonal matrix $\hat{D}_8$ in (5.125) is given by $\hat{D}_8 = \text{diag}\,\{\frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}, \frac{1}{\sqrt{8}}\}$. The generalized signal flow graph for the 8-point DCT-II computation via WHT and its inverse, DCT-III, with highlighted plane rotations is shown in Fig. 5.22. The computation of DCT-II via WHT requires 15 floating-point multiplications and 39 floating-point additions.

The signal flow graph in Fig. 5.22 involves five plane rotations with 3 different rotation angles, specifically,

$$G_{-\frac{\pi}{8}}, \quad G_{-\frac{\pi}{16}} \quad \text{and} \quad G_{-\frac{3\pi}{16}}.$$

In Table 5.20 are summarized all plane rotations with analytical values of the multipliers p_i, u_i, $i = 0, 1, 2, 3, 4$ together with their dyadic approximations and the computational cost of the corresponding LUL structures in terms of add and shift operations.

The general form of the fast multiplierless 8-point IntDCT-II is shown in Fig. 5.23. The fast IntDCT-III implementation is obtained by reversing the IntDCT-II signal flow graph and taking inverses of the LUL structure. Other configurations of IntDCT-II can be found in [50, 51]. If multipliers p_i and u_i are floating-point numbers, then implementations of the 8-point DCT-II in Figs. 5.22 and 5.23 are equivalent.

In Table 5.21 is summarized the MSE approximation error, performance measures of the fast IntDCT-II compared with the 8-point DCT-II, and the total computational complexity of IntDCT-II and IntDCT-III for specific dyadic approximations of the multipliers from Table 5.20.

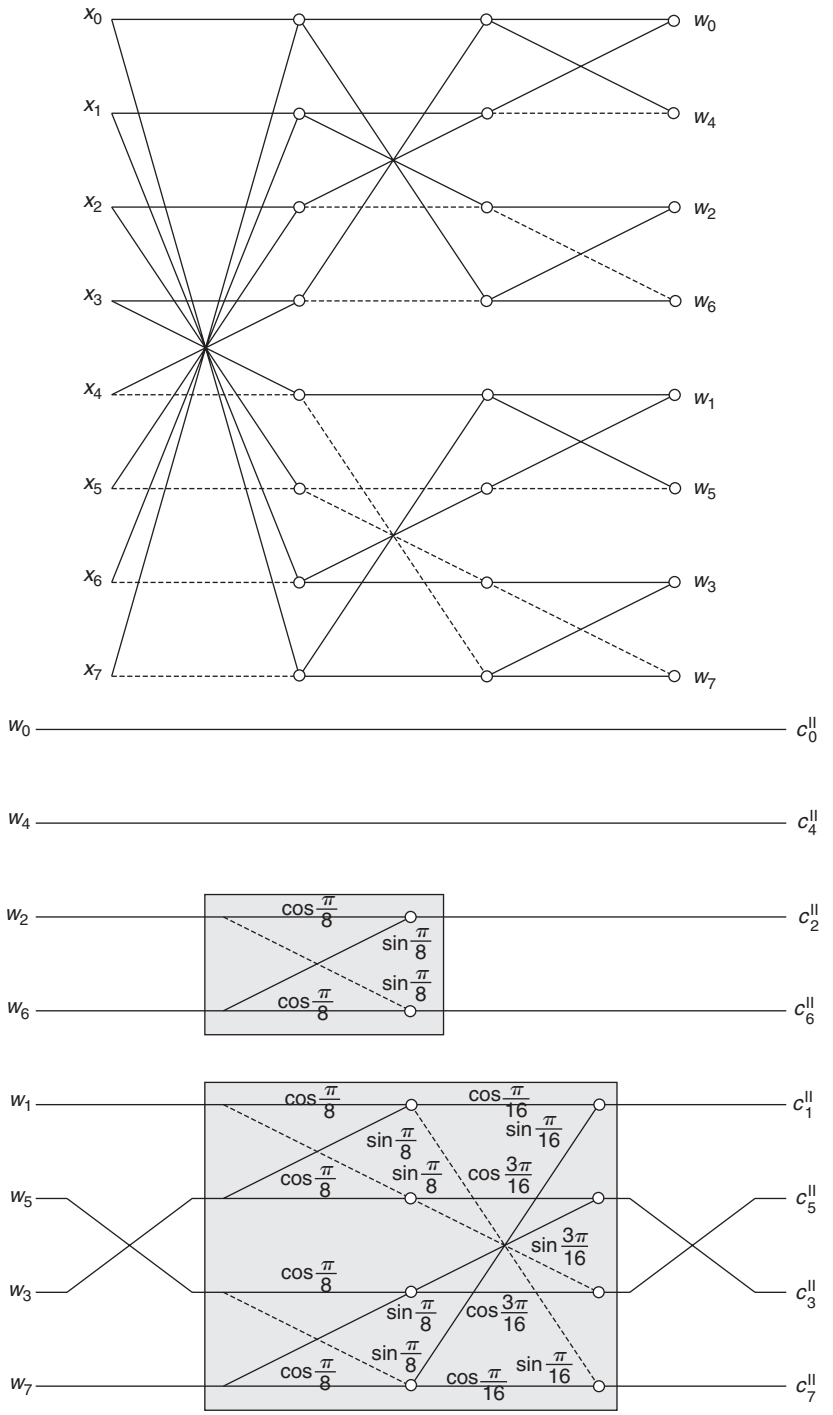


Fig. 5.22. The generalized signal flow graph for the fast 8-point DCT-II computation and its inverse, DCT-III, via WHT with highlighted plane rotations.

Table 5.20. Plane rotations with analytical values of the multipliers, their dyadic approximations and the computational cost of the corresponding LUL structures in IntDCT-II.

Plane rotation	Multiplier	Analytical value	Dyadic approximation	Computational structure	Computational cost (adds + shifts)
$G_{-\frac{\pi}{8}}$	$p_{0,1,2}$	0.198912	$3/16 \approx 0.187500$	LUL	
	$u_{0,1,2}$	0.382683	$3/8 \approx 0.375000$	$\bar{G}_{-\frac{\pi}{8}}(3/16, 3/8)$	7A + 6S
$G_{-\frac{5\pi}{16}}$	p_3	0.098491	$3/32 \approx 0.093750$	LUL	
	u_3	0.195090	$3/16 \approx 0.187500$	$\bar{G}_{-\frac{\pi}{16}}(3/32, 3/16)$	6A + 6S
$G_{-\frac{3\pi}{16}}$	p_4	0.303347	$5/16 \approx 0.312500$	LUL	
	u_4	0.555570	$9/16 \approx 0.562500$	$\bar{G}_{-\frac{3\pi}{16}}(5/16, 9/16)$	6A + 6S

5.4.4.9 Relationships among BinDCTs-II, IntDCTs-II and WHT

It is interesting to note that if all multipliers p_i and u_i are set to zero in the fast multiplierless BinDCT-IIC, BinDCT-IIL, BinDCT-IIS and IntDCT-II, then the relationships among BinDCTs-II, IntDCTs-II and WHT become appearent.

Consider the signal flow graph in Fig. 5.16 used for the construction of BinDCT-IIC. If we remove the intermediate plane rotation $\bar{R}_{\frac{\pi}{4}}$, replace all the other plane rotations by butterflies, and insert a permutation as shown by the dashed box in Fig. 5.24, the factorization (5.116) is reduced to a new 8-point fast WHT, which can be tuned into a special BinDCT-IIC.

Similarly, in constructing BinDCT-IIL from the signal flow graph shown in Fig. 5.18, if we delete 2 plane rotations $G_{-\frac{\pi}{16}}$ and $G_{-\frac{3\pi}{16}}$, and add one more butterfly as shown by the dashed box in Fig. 5.25, the factorization (5.119) is reduced to another new 8-point fast WHT.

In the construction of BinDCT-IIS from the signal flow graph shown in Fig. 5.20, if we remove the plane rotation $G_{\frac{\pi}{4}}$ and replace the other plane rotations by butterflies, the factorization (5.122) is reduced to the exact 8-point fast WHT shown in Fig. 5.22.

In the case of IntDCT-II setting all multipliers p_i and u_i in Fig. 5.20 to zero, it is reduced to the 8-point fast WHT. Hence, the proposed BinDCTs-II and IntDCTs-II families can bridge the gap between DCT-II and WHT by increasing the resolution of the dyadic approximations of the multipliers p_i and u_i.

5.4.4.10 Concluding notes

- Based on the relation between DCT-II and DST-II matrices given by (4.11), the fast multiplierless BinDST-II/BinDST-III and IntDST-II/IntDST-III can be obtained.
- The construction of BinDCT-II/BinDST-II and IntDCT-II/IntDST-II and their inverses, BinDCT-III/BinDST-III and IntDCT-III/IntDST-III, can be further

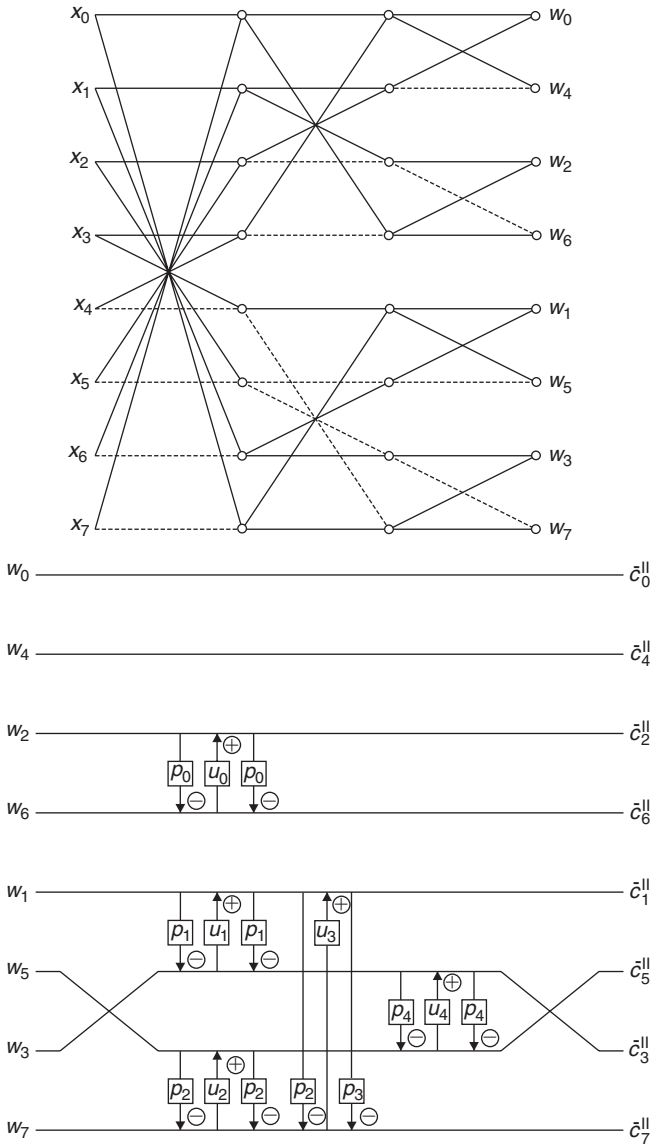


Fig. 5.23. The general form of the fast multiplierless 8-point IntDCT-II.

improved in terms of computational complexity. To reduce the number of add and shift operations, the plane rotations at the end of signal flow graph can be replaced by the DLU structure of G_{φ_i} or H_{φ_i}.

- Extension of the method to $N = 16$ and higher is straightforward. For a given N we need only a sparse factorization of the transform matrix and the corresponding signal flow graph.

Table 5.21. Comparison of the MSE approximation error, performance measures between the fast multiplierless BinDCTs-II, IntDCT-II compared with the 8-point DCT-II, and the total computational complexities of BinDCTs-II and IntDCT-II.

	MSE	C_g	η	Adds/shifts
8-point DCT-II	–	8.82591	93.99119	–
BinDCT-IIC	2.719030e−004	8.81602	93.06690	40/21
BinDCT-IIL	6.958096e−005	8.82228	93.42311	40/21
BinDCT-IIS	8.533458e−005	8.81855	93.52233	41/18
IntDCT-II	5.467468e−005	8.82393	93.75567	54/28

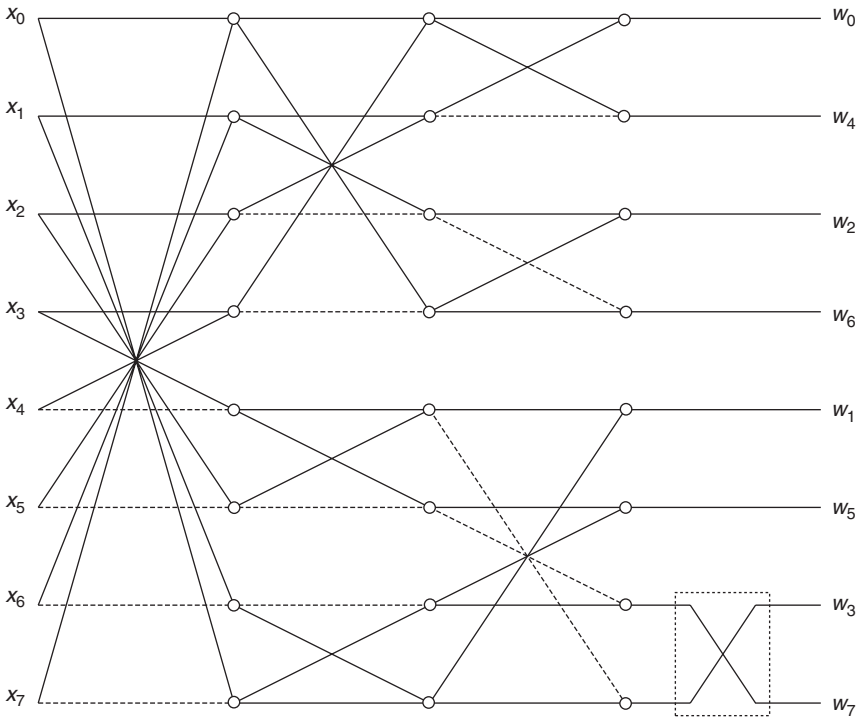


Fig. 5.24. The 8-point fast WHT derived from the factorization (5.116).

- In the implementation of two-dimensional (2-D) BinDCT-II/BinDST-II and IntDCT-II/IntDST-II, and their inverses for transform-based coding applications, the normalization by diagonal elements of $\hat{D}_N$ is reduced to shift operations.
- The dynamic range of BinDCTs-II and IntDCTs-II must be examined carefully in relation to the maximum/minimum values in the input vector. As all elements of BinDCT-II and IntDCT-II have magnitude less than or equal to unity, they can minimize the intermediate dynamic range. Since the absolute sum of the first row of the approximated transform matrix is much greater than that of other rows, the dynamic range of the BinDCTs-II and IntDCTs-II is thus determined by the DC subband

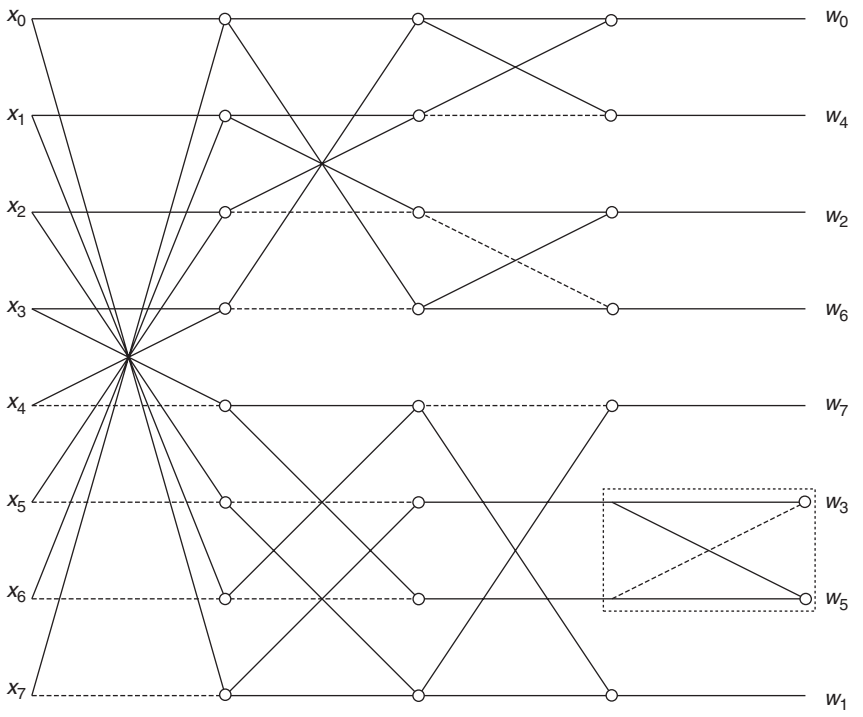


Fig. 5.25. The 8-point fast WHT derived from the factorization (5.119).

(0th coefficient). For 8-point input, a 14-bit representation of the 2-D BinDCT-II and IntDCT-II output coefficients is needed. This way, the implementations of 2-D BinDCTs-II and 2-D IntDCTs-II can well fit into a 16-bit architecture [46].

Compared to the conventional floating-point DCT-II, the BinDCTs-II and IntDCTs-II offer the following advantages [43, 45, 46, 47]:

- Both the forward and inverse transforms have an elegant implementation using only binary add and shift operations. No multiplications are needed.
- BinDCTs-II and IntDCTs-II inherit all desirable DCT-II characteristics such as high coding gain, no DC leakage (for constant signal only the 0th transform coefficient is nonzero), symmetric basis vectors and recursive construction.
- BinDCTs-II and IntDCTs-II also inherit all properties of plane rotation factorizations such as fast implementation, invertible integer-to-integer mapping, in-place computation and low dynamic range. The integer-to-integer mapping with exact reconstruction property is pivotal in transform-based lossless coding and allows for a unifying lossy/lossless coding framework.
- In software implementation the BinDCTs-II and IntDCTs-II is faster than that of floating-point DCT-II. Much higher speed can be achieved in their hardware implementation.

- The multiplierless property of BinDCTs-II and IntDCTs-II allows for an efficient VLSI implementation in terms of chip area and power consumption.
- BinDCTs-II and IntDCTs-II aproximate the DCT-II very closely. Coding strategies designed specifically for the DCT-II can be applied to the BinDCTs-II and IntDCTs-II immediately without any modification.
- The 2-D BinDCTs-II and IntDCTs-II allow 16-bit implementation, enable lossless compression and maintain satisfactory compatibility with the DCT-II.

5.4.4.11 Construction of fast multiplierless BinDCT-IV/BinDST-IV

Finally, as an example, we present the construction of fast multiplierless BinDCT-IV/BinDST-IV for $N = 8$ based on the fast algorithm for DCT-IV computation [21]. Let C_8^{IV} be the DCT-IV matrix. Then, according to the improved sparse factorization (4.83) it can be written as

$$
\begin{aligned}
C_8^{\mathrm{IV}} = D_8 P_8 &
\begin{pmatrix}
\cos\frac{\pi}{32} & \sin\frac{\pi}{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
\sin\frac{\pi}{32} & -\cos\frac{\pi}{32} & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & \cos\frac{5\pi}{32} & \sin\frac{5\pi}{32} & 0 & 0 & 0 & 0 \\
0 & 0 & \sin\frac{5\pi}{32} & -\cos\frac{5\pi}{32} & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & \cos\frac{9\pi}{32} & \sin\frac{9\pi}{32} & 0 & 0 \\
0 & 0 & 0 & 0 & \sin\frac{9\pi}{32} & -\cos\frac{9\pi}{32} & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & \cos\frac{13\pi}{32} & \sin\frac{13\pi}{32} \\
0 & 0 & 0 & 0 & 0 & 0 & \sin\frac{13\pi}{32} & -\cos\frac{13\pi}{32}
\end{pmatrix} \\
& \times \begin{pmatrix} I_4 & I_4 \\ I_4 & -I_4 \end{pmatrix}
\begin{pmatrix}
I_4 & & & & 0 \\
 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 \\
0 & 0 & -1 & 0 & 0 \\
 & 0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
I_2 & I_2 & & & 0 & \\
I_2 & -I_2 & & & & \\
 & & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & 0 & 0 \\
 & & \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & 0 & 0 \\
0 & & 0 & 0 & \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\
 & & 0 & 0 & \sin\frac{\pi}{8} & -\cos\frac{\pi}{8}
\end{pmatrix} \\
& \times \begin{pmatrix}
I_2 & & & & 0 \\
 & \cos\frac{\pi}{4} & \sin\frac{\pi}{4} & & \\
 & \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} & & \\
 & & & I_2 & I_2 \\
0 & & & I_2 & -I_2
\end{pmatrix} H_8,
\end{aligned}
\tag{5.128}
$$

where P_8 is a matrix that permutes odd-indexed components to reversed order, and the matrix H_8 permutes columns of the transform matrix into a Hadamard order. The diagonal

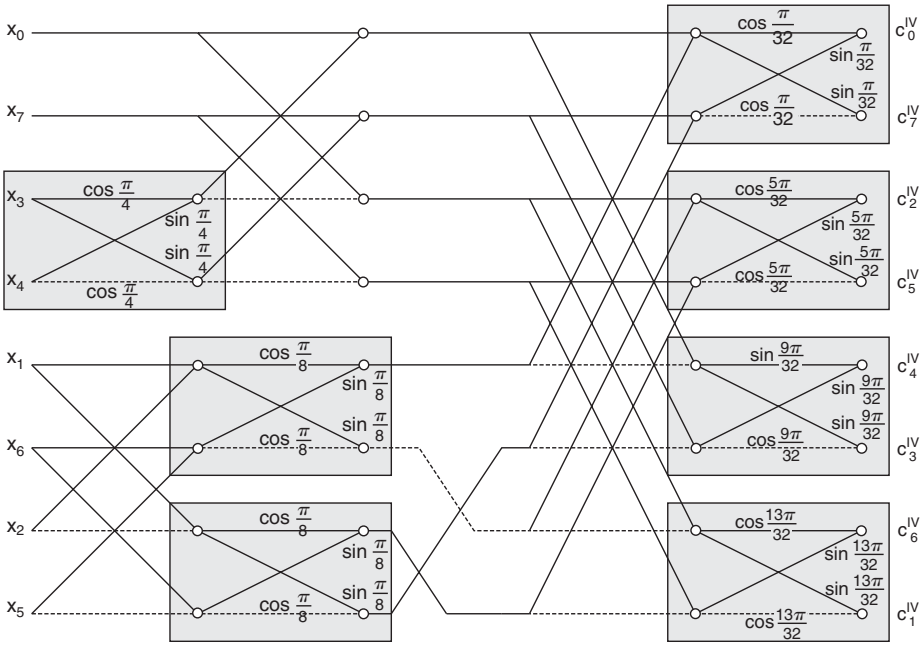


Fig. 5.26. The generalized signal flow graph for the fast 8-point DCT-IV computation based on factorization (5.128) with highlighted plane rotations.

matrix is given by $D_8 = \mathrm{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}\}$. The generalized signal flow graph for the fast 8-point DCT-IV computation based on (5.128) with highlighted plane rotations is shown in Fig. 5.26. The computation of 8-point DCT-IV requires 20 floating-point multiplications and 36 floating-point additions. Comparing Figs. 4.16 with 5.26, we note that the butterfly with multipliers $\cos\frac{\pi}{4}$ in Fig. 4.16 has been replaced by plane rotation with the angle $\frac{\pi}{4}$ in Fig. 5.26.

The signal flow graph in Fig. 5.26 involves 7 plane rotations of the same type, Householder reflections with six different rotation angles, specifically,

$$H_{\frac{\pi}{4}}, \quad H_{\frac{\pi}{8}}, \quad H_{\frac{\pi}{32}}, \quad H_{\frac{5\pi}{32}}, \quad H_{\frac{9\pi}{32}} \quad \text{and} \quad H_{\frac{13\pi}{32}}.$$

Taking into account the relation between H_{φ_i} and G_{φ_i} given by (5.37), each Householder reflection H_{φ_i} can be converted to Givens–Jacobi rotation G_{φ_i} as

$$H_{\varphi_i} = G_{\varphi_i} \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

and G_{φ_i} can be replaced by the modified LUL structure with dyadic approximations of the multipliers p_i, u_i, $i = 0, 1, 2, 3, 4, 5, 6$. Table 5.22 summarizes all plane rotations

Table 5.22. Plane rotations with analytical values of the multipliers, their dyadic approximations and the computational cost of the corresponding LUL structures in BinDCT-IV.

Plane rotation	Multiplier	Analytical value	Dyadic approximation	Computational structure	Computational cost (adds + shifts)
$H_{\frac{\pi}{4}}$	p_0 u_0	0.414214 0.707107	$13/32 \approx 0.406250$ $23/32 \approx 0.718750$	LUL $\bar{G}_{\frac{\pi}{4}}(13/32, 23/32) \times$ $\text{diag}\{1, -1\}$	9A + 8S
$H_{\frac{\pi}{8}}$	$p_{1,2}$ $u_{1,2}$	0.198912 0.382683	$3/16 \approx 0.187500$ $3/8 \approx 0.375000$	LUL $\bar{G}_{\frac{\pi}{8}}(3/16, 3/8) \times$ $\text{diag}\{1, -1\}$	12A + 12S
$H_{\frac{\pi}{32}}$	p_3 u_3	0.049127 0.098017	$3/64 \approx 0.046875$ $3/32 \approx 0.093750$	LUL $\bar{G}_{\frac{\pi}{32}}(3/64, 3/32) \times$ $\text{diag}\{-1, 1\}$	6A + 6S
$H_{\frac{5\pi}{32}}$	p_4 u_4	0.250487 0.471397	$1/4 \approx 0.250000$ $15/32 \approx 0.468750$	LUL $\bar{G}_{\frac{5\pi}{32}}(1/4, 15/32) \times$ $\text{diag}\{-1, 1\}$	4A + 4S
$H_{\frac{9\pi}{32}}$	p_5 u_5	0.472965 0.773010	$15/32 \approx 0.468750$ $3/4 \approx 0.750000$	LUL $\bar{G}_{\frac{9\pi}{32}}(15/32, 3/4) \times$ $\text{diag}\{-1, 1\}$	6A + 6S
$H_{\frac{13\pi}{32}}$	p_6 u_6	0.741651 0.956940	$3/4 \approx 0.750000$ $15/16 \approx 0.937550$	LUL $\bar{G}_{\frac{13\pi}{32}}(3/4, 15/16) \times$ $\text{diag}\{-1, 1\}$	6A + 5S

with analytical values of the multipliers together with their dyadic approximations and the computational cost of the corresponding LUL structures.

The general form of the fast multiplierless 8-point BinDCT-IV is shown in Fig. 5.27. The total computational complexity of BinDCT-IV for specific dyadic approximations of the multipliers shown in Table 5.22 is 59 additions and 41 shifts. Since the DCT-IV matrix is symmetric, the implementation of inverse BinDCT-IV is the same. Based on relation between DCT-IV and DST-IV matrices given by (4.12), the fast multiplierless BinDST-IV transform can be easily obtained. If multipliers p_i and u_i are floating-point numbers then implementations in Figs. 5.26 and 5.27 are equivalent. Any other dyadic approximations of p_i and u_i may be substituted to obtain various configurations of BinDCT-IV with different accuracies and computational costs. In order to reduce the computational cost, plane rotations at the end of the signal flow graph may be replaced by DLU structures.

5.5 Other methods and approaches

The methods discussed in previous sections were originally developed for the integer approximations of DCTs/DSTs. The resulting integer DCTs/DSTs preserve all mathematical properties of the original real-valued transforms and they are efficiently implemented in the integer domain. Besides these, other methods and approaches have been proposed to construct integer DCTs [9–11, 58–65]. Essentially, these methods do not construct integer DCTs directly in the integer domain. The computations are still realized with floating-point

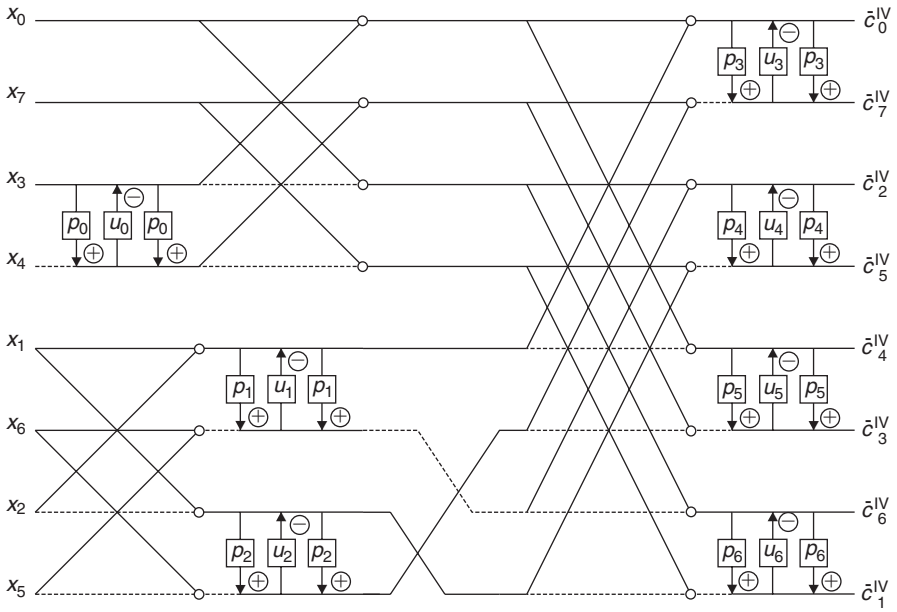


Fig. 5.27. The general form of the fast multiplierless 8-point BinDCT-IV.

arithmetic with a nonlinear operation (*rounding*, *floor* or *ceil*) in the intermediate computational steps. The result is guaranteed to be integer and the invertibility (reversibility) of the transform is preserved. In general, the construction of integer transform is based on:

- A factorization of the transform matrix into a product of invertible matrices with simpler structure (*QR*, *LU* and *PLUS* matrix factorizations).
- A (recursive) factorization of the transform matrix into the product of sparse, orthogonal matrices, where each matrix factor is a block-diagonal matrix with blocks being orthogonal matrices of order 2. Then each 2×2 orthogonal (rotation) matrix is subsequently again factorized and integer approximated, for example, by rounding procedure such that the process is invertible (reversible).

In the following sections principles for the construction of integer transforms such as a lossless DCT [58], DCT with reversible integer mapping [9–11], invertible integer DCTs [59–61], reversible DCTs [62, 63] and square wave transform based on DCT-II [65] are described in detail.

5.5.1 Lossless DCT

The concept of lossless DCT-II (LDCT-II) has been introduced in Ref. [58]. The LDCT-II is a modification of the DCT-II that produces integer transform coefficients. The principle behind the LDCT-II is very simple. It is based on the factorization of DCT-II matrix

C_N^{II} into the following product:

$$C_N^{\mathrm{II}} = D\,P_1 L\,U P_2, \tag{5.129}$$

where P_1 and P_2 are permutation matrices, D is a diagonal matrix for scaling transform coefficients, and L, U are respectively a unit lower and unit upper triangular matrix. For a given integer input data vector $\mathbf{x}$ the LDCT-II computes the integer transform coefficient vector $\bar{\mathbf{c}}^{\mathrm{II}}$ as

$$\bar{\mathbf{c}}^{\mathrm{II}} = P_1 \lfloor L \lfloor U\,P_2\,\mathbf{x}^{\mathrm{T}} \rfloor \rfloor, \tag{5.130}$$

where $\lfloor \cdot \rfloor$ denotes the rounding to nearest integer. Thus, the LDCT-II is implemented in floating-point arithmetic with the results immediately rounded to the nearest integer. Since inverses of L, U matrices exist (L^{-1} and U^{-1} are of the same type), the original input data vector can be losslessly recovered from $\bar{\mathbf{c}}^{\mathrm{II}}$ by the inverse LDCT-II. The computational complexity of the forward and inverse LDCT-II is of order a matrix–vector multiplication. The maximum error (truncation error introduced by rounding operator) is bounded with the bound being independent of the range of input data vector.

A general matrix factorization theory for reversible integer mapping of an invertible linear transform is presented in Refs. [9–11]. Factorizing the transform matrix into a product of (unit) triangular matrices (*LU* and *PLUS* factorizations) and introducing rounding operator is the basic approach for implementing the transform by reversible integer mapping. The advantages of integer implementation of invertible linear transform are:

- integer-to-integer mapping,
- perfect reconstruction,
- in-place computation.

In view of linear algebra and matrix computations the DCT/DST matrices possess nice mathematical properties such as linearity, orthogonality/orthonormality, symmetry of the basis vectors, eigenorthogonality/eigenorthonormality and recursiveness. In order to obtain an implementation of DCT/DST with reversible integer mapping, we utilize the results of linear algebra only: *QR*, *LU* and *PLUS* matrix factorizations. Such factorizations allow to factorize a transform matrix to be written as products of structurally simpler matrices to efficiently realize subsequent computational steps.

Consider the 8-point DCT-II defined by the matrix C_8^{II}. The direct analytical derivation of *QR*, *LU* and *PLUS* factorizations for the matrix C_8^{II} can be a relatively complex procedure. In order to simplify the analytical derivation of *QR*, *LU* and *PLUS* for C_8^{II} we exploit the EOT factorization defined by (4.17). According to the EOT factorization, the matrix C_8^{II} can be recursively reduced to the butterfly matrices and DCT-II and DCT-IV matrices of lower orders. Finding *QR*, *LU* and *PLUS* factorizations of the lower-order DCT-II and DCT-IV matrices and substituting them into the EOT factorization results in the regular computational structures for 8-point DCT-II computation with reversible integer mapping. With respect to [58] they will be called *QR*-, *LU*- and *PLUS*-based LDCTs-II. Although the approach will be illustrated for the 8-point DCT-II, in general, it can be applied to any DCT and DST.

5.5.1.1 EOT matrix factorization of $\hat{C}_8^{II}$

Let $\hat{C}_8^{\mathrm{II}}$ be the DCT-II matrix with its rows rearranged so that the upper half consists of even-indexed rows followed by the lower half of odd-indexed rows, both in natural order. Then, according to the EOT factorization defined by (4.17), the matrix $\hat{C}_8^{\mathrm{II}}$ can be written as

$$\hat{C}_8^{\mathrm{II}} = \frac{1}{2}\begin{pmatrix} \bar{C}_4^{\mathrm{II}} & \bar{C}_4^{\mathrm{II}} J_4 \\ \bar{C}_4^{\mathrm{IV}} & -\bar{C}_4^{\mathrm{IV}} J_4 \end{pmatrix} = \frac{1}{2}\begin{pmatrix} \bar{C}_4^{\mathrm{II}} & 0 \\ 0 & \bar{C}_4^{\mathrm{IV}}\, J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.131}$$

where

$$\bar{C}_4^{\mathrm{II}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{pmatrix},$$

$$\bar{C}_4^{\mathrm{IV}} J_4 = \begin{pmatrix} \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} \\ -\sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \end{pmatrix}. \tag{5.132}$$

The eigenorthonormal matrices C_4^{II} and $C_4^{\mathrm{IV}} J_4$ (see Section 4.3) are given by

$$C_4^{\mathrm{II}} = \frac{\sqrt{2}}{2}\begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \cos\frac{\pi}{8} & \sin\frac{\pi}{8} & -\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \cos\frac{\pi}{8} & -\sin\frac{\pi}{8} \end{pmatrix},$$

$$C_4^{\mathrm{IV}} J_4 = \frac{\sqrt{2}}{2}\begin{pmatrix} \sin\frac{\pi}{16} & \sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & \cos\frac{\pi}{16} \\ -\sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \cos\frac{3\pi}{16} & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{3\pi}{16} \\ -\cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & -\sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \end{pmatrix}$$

$$= \frac{\sqrt{2}}{2}\begin{pmatrix} \sin\frac{\pi}{16} & \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) & \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}) & \cos\frac{\pi}{16} \\ -\frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}) \\ \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}) & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) \\ -\cos\frac{\pi}{16} & \frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}) & -\frac{\sqrt{2}}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) & \sin\frac{\pi}{16} \end{pmatrix}. \tag{5.133}$$

Comparing (5.132) and (5.133), we have

$$\bar{C}_4^{\mathrm{II}} = \frac{\sqrt{2}}{2} C_4^{\mathrm{II}}, \quad \bar{C}_4^{\mathrm{IV}} J_4 = \frac{\sqrt{2}}{2} C_4^{\mathrm{IV}} J_4. \tag{5.134}$$

Thus,

$$\hat{C}_8^{\mathrm{II}} = \frac{\sqrt{2}}{2} \begin{pmatrix} C_4^{\mathrm{II}} & 0 \\ 0 & C_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}. \tag{5.135}$$

Now, since C_4^{II} and $C_4^{\mathrm{IV}} J_4$ have $\det(C_4^{\mathrm{II}}) = \det(C_4^{\mathrm{IV}} J_4) = +1$, they must have *QR*, *LU* and *PLUS* factorizations. Finding *QR*, *LU* and *PLUS* factorizations for C_4^{II} and $C_4^{\mathrm{IV}} J_4$ and substituting them into the modified EOT factorization (5.135) we obtain regular computational structures for the 8-point DCT-II with reversible integer mapping.

The DCT-II matrix $\hat{C}_8^{\mathrm{II}}$ possesses recursive property. Therefore, the matrix C_4^{II} in (5.135) can be again factorized according to the EOT factorization. Similarly, let $\hat{C}_4^{\mathrm{II}}$ be the matrix with its rows rearranged so that the upper half consists of even-indexed rows followed by the lower half of odd-indexed rows, both in natural order. Then, it can be written as

$$\hat{C}_4^{\mathrm{II}} = \frac{\sqrt{2}}{2} \begin{pmatrix} C_2^{\mathrm{II}} & C_2^{\mathrm{II}} J_2 \\ C_2^{\mathrm{IV}} & -C_2^{\mathrm{IV}} J_2 \end{pmatrix} = \frac{\sqrt{2}}{2} \begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & C_2^{\mathrm{IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix}, \tag{5.136}$$

where

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix} = \begin{pmatrix} \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \\ \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} J_2 = G_{-\frac{\pi}{4}} J_2,$$

$$C_2^{\mathrm{IV}} J_2 = \begin{pmatrix} \sin\frac{\pi}{8} & \cos\frac{\pi}{8} \\ -\cos\frac{\pi}{8} & \sin\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix} = G_{-\frac{3\pi}{8}}, \tag{5.137}$$

where $G_{-\frac{\pi}{4}}$ and $G_{-\frac{3\pi}{8}}$ are Givens–Jacobi rotations and $\det(G_{-\frac{\pi}{4}}) = \det(G_{-\frac{3\pi}{8}}) = +1$. Therefore, the modified EOT factorization (5.135) can be further simplified and improved in terms of computational complexity and structural simplicity.

5.5.1.2 QR-based factorization of DCT matrices

QR factorization of a real square nonsingular matrix A of order N is stated by Theorem 5.1 and Corollary 5.1 in Section 5.2.7, where Q is an orthogonal matrix and R is an upper triangular matrix. We note that the *QR* factorization is also discussed in Appendix A.3

(see equations (A.27)–(A.34)). The *QR* process is useful only in the determination of the elementary Givens–Jacobi rotation matrices which are easily implemented via orthogonal butterflies in the flow diagram. Actual orthogonal matrix Q is not used.

The *QR* factorization algorithm [1] consists of successive premultiplications (multiplications on the left) of the matrix A by elementary Givens–Jacobi rotation matrices G_{ij}, $i<j$, defined by (5.11). In one step of the algorithm to null the element $a_{ji}\neq 0$ of A under principal diagonal, the values of c and s are chosen such that

$$s=-\frac{a_{ji}}{\sqrt{a_{ii}^2+a_{ji}^2}},\qquad c=\frac{a_{ii}}{\sqrt{a_{ii}^2+a_{ji}^2}},\qquad i=1,2,\ldots,N \text{ and } j=i+1,\ldots,N, \tag{5.138}$$

and new values of elements in ith and jth rows of A are given by

$$\begin{aligned} a_{ik}^{(1)} &= ca_{ik}-sa_{jk},\\ a_{jk}^{(1)} &= sa_{ik}+ca_{jk},\quad k=1,2,\ldots,N. \end{aligned} \tag{5.139}$$

For $k=i$ from (5.138) and (5.139) it follows that

$$a_{ii}^{(1)}=\sqrt{a_{ii}^2+a_{ji}^2}>0,\quad a_{ji}^{(1)}=0.$$

Since the matrices C_4^{II} and $C_4^{\mathrm{IV}}J_4$ in (5.133) are eigenorthonormal, according to Corollary 5.2 of Theorem 5.1 in Section 5.2.7 they can be factorized into the product only of at most 6 elementary Givens–Jacobi rotation matrices G_{ij} as follows:

$$\begin{aligned} C_4^{\mathrm{II}} &= Q_4^{(1)}R_4,\\ C_4^{\mathrm{IV}}J_4 &= Q_4^{(2)}R_4, \end{aligned} \tag{5.140}$$

where

$$\begin{aligned} Q_4^{(1)} &= G_{12}^{-1}(\varphi_1)G_{13}^{-1}(\varphi_2)G_{14}^{-1}(\varphi_3)G_{23}^{-1}(\varphi_4)G_{24}^{-1}(\varphi_5)G_{34}^{-1}(\varphi_6),\\ Q_4^{(2)} &= G_{12}^{-1}(\psi_1)G_{13}^{-1}(\psi_2)G_{14}^{-1}(\psi_3)G_{23}^{-1}(\psi_4)G_{24}^{-1}(\psi_5)G_{34}^{-1}(\psi_6), \end{aligned} \tag{5.141}$$

and $R_4=I_4$ is the identity matrix. In fact, by the numerical procedure defined by (5.138) and (5.139) we obtain the following factorizations of C_4^{II} and $C_4^{\mathrm{IV}}J_4$ matrices, where $Q_4^{(1)}$

and $Q_4^{(2)}$ matrices are products of the following elementary Givens–Jacobi rotation matrices

$$Q_4^{(1)} = \begin{pmatrix} 0.60778 & -0.79410 & 0 & 0 \\ 0.79410 & 0.60778 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.85455 & 0 & -0.51938 & 0 \\ 0 & 1 & 0 & 0 \\ 0.51938 & 0 & 0.85455 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 0.96269 & 0 & 0 & -0.27060 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.27060 & 0 & 0 & 0.96269 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.31666 & 0.94854 & 0 \\ 0 & -0.94854 & -0.31666 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.73451 & 0 & 0.67860 \\ 0 & 0 & 1 & 0 \\ 0 & -0.67860 & 0 & 0.73451 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.38268 & -0.92388 \\ 0 & 0 & 0.92388 & -0.38268 \end{pmatrix},$$

and

$$Q_4^{(2)} = \begin{pmatrix} 0.76278 & -0.64666 & 0 & 0 \\ 0.64666 & 0.76278 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0.91797 & 0 & -0.39664 & 0 \\ 0 & 1 & 0 & 0 \\ 0.39664 & 0 & 0.91797 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 0.99044 & 0 & 0 & -0.13795 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0.13795 & 0 & 0 & 0.99044 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -0.52879 & 0.84875 & 0 \\ 0 & -0.84875 & -0.52879 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.91797 & 0 & 0.39664 \\ 0 & 0 & 1 & 0 \\ 0 & -0.39664 & 0 & 0.91797 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -0.76278 & -0.64666 \\ 0 & 0 & 0.64666 & -0.76278 \end{pmatrix}.$$

However, such factorization procedure is more convenient rather for purposes of numerical analysis. We alternatively derive QR factorizations of C_4^{II} and C_4^{IV} J_4 matrices by the analytical procedure with proper choice of premultiplied elementary Givens–Jacobi rotation matrices G_{ij}.

For the C_4^{II} matrix we get the following factorization:

$$
\begin{aligned}
C_4^{\mathrm{II}} &= \begin{pmatrix} \cos\frac{\pi}{4} & 0 & -\sin\frac{\pi}{4} & 0 \\ 0 & 1 & 0 & 0 \\ \sin\frac{\pi}{4} & 0 & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{3\pi}{8} & 0 & \sin\frac{3\pi}{8} \\ 0 & 0 & 1 & 0 \\ 0 & -\sin\frac{3\pi}{8} & 0 & \cos\frac{3\pi}{8} \end{pmatrix} \\
&\quad \times \begin{pmatrix} \cos\frac{\pi}{4} & 0 & 0 & -\sin\frac{\pi}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \sin\frac{\pi}{4} & 0 & 0 & \cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\frac{\pi}{4} & \sin\frac{\pi}{4} & 0 \\ 0 & -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\
&= G_{13}\Big(\frac{\pi}{4}\Big) G_{24}\Big(-\frac{3\pi}{8}\Big) G_{14}\Big(\frac{\pi}{4}\Big) G_{23}\Big(-\frac{\pi}{4}\Big) R_4 \\
&= G_{24}\Big(-\frac{3\pi}{8}\Big) G_{13}\Big(\frac{\pi}{4}\Big) G_{23}\Big(-\frac{\pi}{4}\Big) G_{14}\Big(\frac{\pi}{4}\Big) R_4 \\
&= Q_4 R_4, \qquad (5.142)
\end{aligned}
$$

and for the $C_4^{\mathrm{IV}} J_4$ matrix we get the factorization:

$$
\begin{aligned}
C_4^{\mathrm{IV}} J_4 &= \begin{pmatrix} \cos\frac{7\pi}{16} & 0 & 0 & \sin\frac{7\pi}{16} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\frac{7\pi}{16} & 0 & 0 & \cos\frac{7\pi}{16} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{16} & -\sin\frac{\pi}{16} & 0 \\ 0 & \sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \\
&\quad \times \begin{pmatrix} \cos\frac{\pi}{4} & 0 & 0 & \sin\frac{\pi}{4} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\sin\frac{\pi}{4} & 0 & 0 & \cos\frac{\pi}{4} \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 & 0 \\ \sin\frac{\pi}{4} & \cos\frac{\pi}{4} & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ 0 & 0 & \sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} \\
&\quad \times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & -\cos\frac{\pi}{4} & -\sin\frac{\pi}{4} & 0 \\ 0 & \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \\
&= G_{14}\Big(-\frac{7\pi}{16}\Big)\Big[-G_{23}\Big(-\frac{\pi}{16}\Big)\Big] P_4\, G_{14}\Big(-\frac{\pi}{4}\Big) G_{12}\Big(\frac{\pi}{4}\Big) G_{34}\Big(\frac{\pi}{4}\Big)\Big[-G_{23}\Big(-\frac{\pi}{4}\Big)\Big] R_4 \\
&= \Big[-G_{23}\Big(-\frac{\pi}{16}\Big)\Big] G_{14}\Big(-\frac{7\pi}{16}\Big) P_4 G_{14}\Big(-\frac{\pi}{4}\Big) G_{34}\Big(\frac{\pi}{4}\Big) G_{12}\Big(\frac{\pi}{4}\Big)\Big[-G_{23}\Big(-\frac{\pi}{4}\Big)\Big] R_4 \\
&= Q_4 R_4, \qquad (5.143)
\end{aligned}
$$

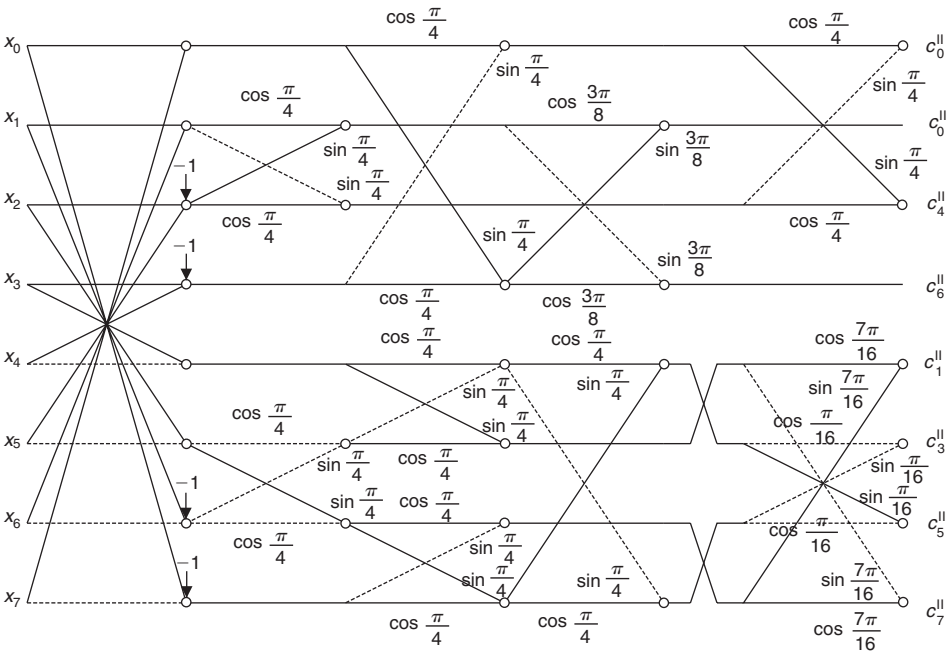


Fig. 5.28. *QR*-based structure for the 8-point DCT-II computation.

where $P_4 = \text{diag}\{J_2, J_2\}$ is a permutation matrix and $R_4 = \text{diag}\{1, 1, -1, -1\}$ is a diagonal matrix. Comparing (5.142) and (5.143) with (5.140) the following differences are evident. *QR* factorizations of C_4^{II} and $C_4^{\text{IV}} J_4$ eigenorthonormal matrices given by (5.140) obtained by the numerical procedure in both cases involve 6 elementary Givens–Jacobi rotation matrices with 6 different angles. *QR* factorization of C_4^{II} (5.142) derived by the analytical procedure involves only 4 elementary Givens–Jacobi rotation matrices with 2 different angles, while *QR* factorization of $C_4^{\text{IV}} J_4$ (5.143) involves 6 elementary Givens–Jacobi rotation matrices with 3 different angles including the permutation matrix P_4. The numerical approach produces I_4 for the R_4 factor whereas the analytical approach produces $R_4 = \text{diag}\{1, 1, -1, -1\}$, both of which have unit determinants. We note that some elementary Givens–Jacobi rotation matrices in (5.142) and (5.143) are commutative.

Substituting analytically derived *QR* factorizations of C_4^{II} and $C_4^{\text{IV}} J_4$ matrices into the modified EOT factorization of $\hat{C}_8^{\text{II}}$ defined by (5.135) we obtain the regular *QR*-based structure for the 8-point DCT-II computation which is shown in Fig. 5.28. Normalization factors for the output transform coefficients are defined by the diagonal matrix $\hat{D}_8 = \text{diag}\{\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$.

As the matrix C_4^{II} in (5.135) can be again factorized according to EOT factorization defined by (5.136), where $C_2^{\text{II}} = G_{-\frac{\pi}{4}} J_2$ and $C_2^{\text{IV}} J_2 = G_{-\frac{3\pi}{8}}$ are Givens–Jacobi rotations, the *QR*-based structure in Fig. 5.28 can be simplified and improved replacing *QR* factorization of C_4^{II} by the EOT factorization of $\hat{C}_4^{\text{II}}$ defined by (5.136) and (5.137). The

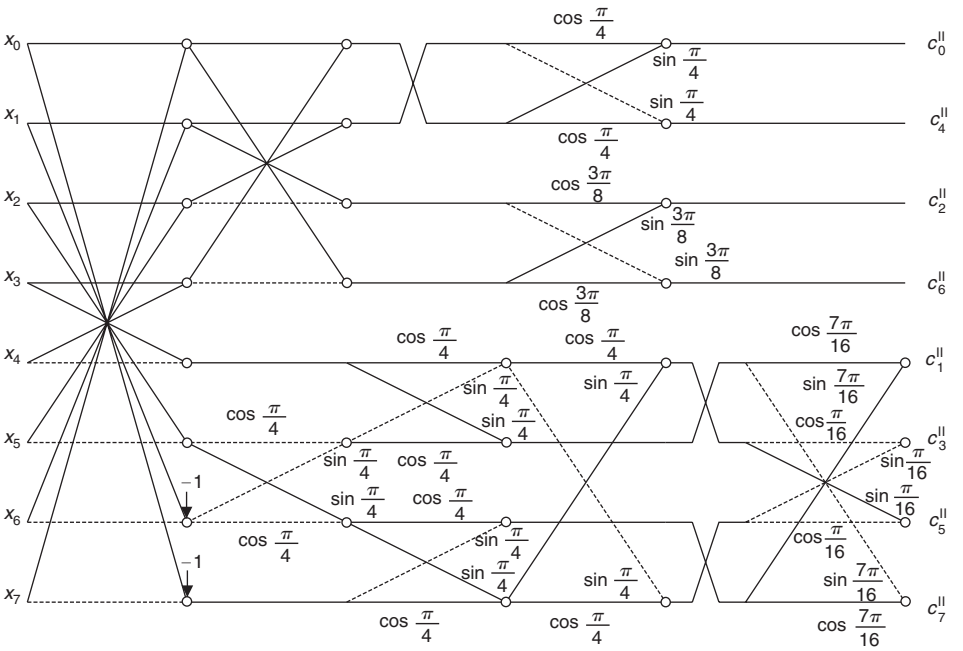


Fig. 5.29. The simplified and improved *QR*-based structure for the 8-point DCT-II computation.

final simplified and improved *QR*-based structure for the 8-point DCT-II computation is shown in Fig. 5.29. Normalization factors for the output transform coefficients are defined by $\hat{D}_8 = \text{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$. The computational complexity of the final *QR*-based structure is 19 floating-point multiplications and 31 floating-point additions provided that the computational cost of each Givens–Jacobi rotation with the angle $\frac{\pi}{4}$ is reduced to 2 multiplications and 2 additions.

EOT factorization of the transform matrix $\hat{C}_8^{II}$ applied recursively in combination with *QR* factorizations of lower-order matrices C_2^{II}, $C_2^{IV}J_2$ and $C_4^{IV}J_4$ provides the simplified and improved *QR*-based structure shown in Fig. 5.29. It consists of butterflies that do not destroy the integer-to-integer mapping, and Givens–Jacobi rotations G_φ or $\pm G_{-\varphi}$. We know that Givens–Jacobi rotation G_φ can be factorized into the product of Gauss elementary matrices that are unit lower and unit upper triangular matrices defining LUL or ULU structures. In fact, the matrix factorizations and corresponding computational structures of G_φ are key mathematical tools in constructing the multiplierless approximations of fast transforms which are described in Section 5.4.4. In the *QR*-based structure if we replace each Givens–Jacobi rotation by its proper LUL structure with approximation of floating-point multipliers such as dyadic rationals, we obtain the multiplierless approximation of 8-point DCT-II with reversible integer mapping called the *QR*-based LDCT-II.

QR-based structure in Fig. 5.29 involves 8 Givens–Jacobi rotations with 4 different angles, specifically,

$$G_{-\frac{\pi}{4}}, \quad -G_{-\frac{\pi}{4}}, \quad G_{\frac{\pi}{4}}, \quad G_{-\frac{3\pi}{8}}, \quad -G_{-\frac{\pi}{16}} \quad \text{and} \quad G_{-\frac{7\pi}{16}}.$$

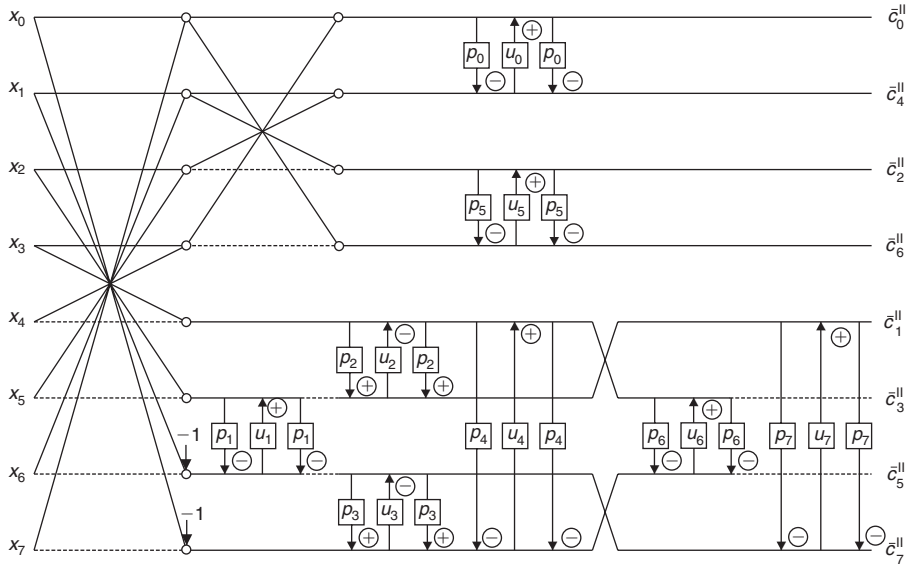


Fig. 5.30. The general form of the forward multiplierless 8-point *QR*-based LDCT-II.

It is sufficient to replace each Givens–Jacobi rotation by the proper LUL structure with dyadic approximation of the multipliers p_i and u_i, $i = 0, 1, \ldots, 7$. The general form of the forward multiplierless 8-point *QR*-based LDCT-II is shown in Fig. 5.30. The normalization factors for output LDCT-II coefficients are defined by $\hat{D}_8 = \text{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$.

If multipliers p_i and u_i are floating-point numbers then the implementations of the 8-point DCT-II in Figs. 5.29 and 5.30 are equivalent. In Table 5.23 are summarized all Givens–Jacobi rotations with analytical values of the multipliers p_i and u_i together with their dyadic approximations and computational cost of the corresponding LUL structures in terms of the number of add and shift operations. The total computational complexity of multiplierless 8-point *QR*-based LDCT-II for the specific dyadic approximations of multipliers shown in Table 5.23 is 62 additions and 49 shifts. The inverse multiplierless 8-point *QR*-based LDCT-II, LDCT-III, is obtained by reversing LDCT-II computational structure and taking inverses of the LUL structures.

5.5.1.3 LU-based factorization of DCT matrices

LU factorization of a real square nonsingular matrix A of order N is stated by Theorem 5.2 in Section 5.2.7, where $L = \{l_{ij}\}$, $l_{ij} = 0$ for $j > i$, is a lower triangular matrix, and $U = \{u_{ij}\}$, $u_{ij} = 0$ for $i > j$, is an upper triangular matrix. We note that the *LU* factorization is also discussed in Appendix A.3 (see equations (A.18)–(A.24)).

Generally, the sufficient condition for the matrix A to have *LU* factorization is to be nonsingular. If the matrix A has *LU* factorization, then the elements of $A = \{a_{ij}\}$, $i, j = 1, 2, \ldots, N$

Table 5.23. Givens–Jacobi rotations with analytical values of the multipliers, their dyadic approximations and computational cost of the corresponding LUL structures in multiplierless 8-point *QR*-based LDCT-II.

Givens–Jacobi rotation	Multiplier	Analytical value	Dyadic approximation	Computational cost of LUL structure (adds + shifts)
$G_{\frac{\pi}{4}}(\pm)G_{-\frac{\pi}{4}}$	$p_{0,1,2,3,4}$	0.414214	$7/16 \approx 0.437500$	
	$u_{0,1,2,3,4}$	0.707107	$3/4 \approx 0.750000$	6A + 6S
$G_{-\frac{3\pi}{16}}$	p_5	0.668179	$5/8 \approx 0.625000$	
	u_5	0.923880	$15/16 \approx 0.937500$	6A + 5S
$G_{-\frac{\pi}{16}}$	p_6	0.098491	$3/32 \approx 0.093750$	
	u_6	0.195090	$3/16 \approx 0.187500$	7A + 8S
$G_{-\frac{7\pi}{16}}$	p_7	0.820679	$13/16 \approx 0.093750$	
	u_7	0.980785	1	7A + 6S

can be expressed in terms of the *LU* matrix product as

$$a_{ij} = \begin{cases} \sum_{k=1}^{j} l_{ik}u_{kj}, & \text{if } i \geq j, \\ \sum_{k=1}^{i} l_{ik}u_{kj}, & \text{if } i < j. \end{cases} \tag{5.144}$$

Equation (5.144) defines a system of linear equations. If we prescribe values for the diagonal elements L or U to be 1*s*, i.e., $l_{ii} = 1$ or $u_{ii} = 1$, respectively, then the system (5.144) has the unique solutions and hence, such *LU* factorizations of A are uniquely determined. In the following we prefer the elements on principal diagonal of U to be $u_{ii} = 1$, and U will be the unit triangular matrix.

Consider the eigenorthonormal matrices C_4^{II} and $C_4^{\mathrm{IV}}J_4$ given by (5.133). Then according to (5.144) the *LU* matrix product is given by

$$\begin{aligned} C_4^{\mathrm{II}} &= C_4^{\mathrm{IV}}J_4 = LU \\ &= \begin{pmatrix} l_{11} & l_{11}u_{12} & l_{11}u_{13} & l_{11}u_{14} \\ l_{21} & l_{21}u_{12} + l_{22} & l_{21}u_{13} + l_{22}u_{23} & l_{21}u_{14} + l_{22}u_{24} \\ l_{31} & l_{31}u_{12} + l_{32} & l_{31}u_{13} + l_{32}u_{23} + l_{33} & l_{31}u_{14} + l_{32}u_{24} + l_{33}u_{34} \\ l_{41} & l_{41}u_{12} + l_{42} & l_{41}u_{13} + l_{42}u_{23} + l_{43} & l_{41}u_{14} + l_{42}u_{24} + l_{43}u_{34} + l_{44} \end{pmatrix}. \end{aligned} \tag{5.145}$$

Comparing the corresponding matrix elements in (5.145) and solving the system of linear equations for the C_4^{II} we obtain

$$C_4^{\mathrm{II}} = LU = \begin{pmatrix} \frac{1}{2} & 0 & 0 & 0 \\ \frac{\sqrt{2}}{2}\cos\frac{\pi}{8} & -\sin\frac{\pi}{8} & 0 & 0 \\ \frac{1}{2} & -1 & \cot\frac{\pi}{8}-1 & 0 \\ \frac{\sqrt{2}}{2}\sin\frac{\pi}{8} & -\cos\frac{\pi}{8} & \frac{1}{\sin\frac{\pi}{8}} & l_{44} \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & \cot\frac{\pi}{8} & \sqrt{2}\cot\frac{\pi}{8} \\ 0 & 0 & 1 & \frac{\sqrt{2}\cot\frac{\pi}{8}}{\cot\frac{\pi}{8}-1} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \tag{5.146}$$

where

$$l_{44} = \frac{\sqrt{2}[\sin\frac{\pi}{8}(1-2\cot\frac{\pi}{8}) - \cos\frac{\pi}{8}\cot\frac{\pi}{8}]}{\cot\frac{\pi}{8}-1}.$$

Since $\det(C_4^{\mathrm{II}}) = +1$ we have $\det(C_4^{\mathrm{II}}) = \det(LU) = \det(L)\det(U) = +1$ and, therefore, $\det(L) = +1$. LU factorization of C_4^{II} given by (5.146) can be converted to DLU scaling up the matrix L by a diagonal matrix D. Then L will have the unit-diagonal elements. This is just the approach used in construction of the 8-point LDCT-II in Ref. [58].

Similarly, comparing the corresponding matrix elements in (5.145) and solving the system of linear equations for the $C_4^{\mathrm{IV}}J_4$ we obtain

$$\begin{aligned} C_4^{\mathrm{IV}} J_4 &= LU \\ &= \begin{pmatrix} \frac{\sqrt{2}}{2}\sin\frac{\pi}{16} & 0 & 0 & 0 \\ -\frac{1}{2}(\cos\frac{\pi}{16}-\sin\frac{\pi}{16}) & l_{22} & 0 & 0 \\ \frac{1}{2}(\cos\frac{\pi}{16}+\sin\frac{\pi}{16}) & l_{32} & l_{33} & 0 \\ -\frac{\sqrt{2}}{2}\cos\frac{\pi}{16} & l_{42} & l_{43} & l_{44} \end{pmatrix} \begin{pmatrix} 1 & \frac{\sqrt{2}}{2}(\cot\frac{\pi}{16}-1) & \frac{\sqrt{2}}{2}(\cot\frac{\pi}{16}+1) & \cot\frac{\pi}{16} \\ 0 & 1 & u_{23} & u_{24} \\ 0 & 0 & 1 & u_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}, \end{aligned} \tag{5.147}$$

where

$$\begin{aligned} l_{22} &= -\frac{\sqrt{2}}{2}\left[\cos\frac{\pi}{16} - \frac{1}{2}\left(\cos\frac{\pi}{16}-\sin\frac{\pi}{16}\right)\left(\cot\frac{\pi}{16}-1\right)\right], \\ l_{32} &= \frac{\sqrt{2}}{2}\left[\sin\frac{\pi}{16} - \frac{1}{2}\left(\cos\frac{\pi}{16}+\sin\frac{\pi}{16}\right)\left(\cot\frac{\pi}{16}-1\right)\right], \\ l_{33} &= -\frac{\sqrt{2}}{2}\left[\cos\frac{\pi}{16} + \frac{1}{2}\left(\cos\frac{\pi}{16}+\sin\frac{\pi}{16}\right)\left(\cot\frac{\pi}{16}+1\right)\right] - l_{32}u_{23}, \end{aligned}$$

$$l_{42} = \frac{1}{2}\left[\left(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}\right) + \cos\frac{\pi}{16}\left(\cot\frac{\pi}{16} - 1\right)\right],$$
$$l_{43} = -\frac{1}{2}\left[\left(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}\right) - \cos\frac{\pi}{16}\left(\cot\frac{\pi}{16} + 1\right)\right] - l_{42}u_{23},$$
$$l_{44} = \frac{\sqrt{2}}{2}\left[\sin\frac{\pi}{16} + \cos\frac{\pi}{16}\cot\frac{\pi}{16}\right] - l_{42}u_{24} - l_{43}u_{34},$$

and

$$u_{23} = \frac{\sin\frac{\pi}{16} - \frac{1}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16})(\cot\frac{\pi}{16} + 1)}{\cos\frac{\pi}{16} - \frac{1}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16})(\cot\frac{\pi}{16} - 1)},$$
$$u_{24} = -\frac{\sqrt{2}}{2}\frac{[(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}) + (\cos\frac{\pi}{16} - \sin\frac{\pi}{16})\cot\frac{\pi}{16}]}{[\cos\frac{\pi}{16} - \frac{1}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16})(\cot\frac{\pi}{16} - 1)]},$$
$$u_{34} = \frac{\frac{1}{2}[(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) - (\cos\frac{\pi}{16} + \sin\frac{\pi}{16})\cot\frac{\pi}{16}] - l_{32}u_{24}}{l_{33}}.$$

Analogously, since $\det(C_4^{\mathrm{IV}}J_4) = +1$ we have $\det(C_4^{\mathrm{IV}}J_4) = \det(LU) = \det(L)\det(U) = +1$ and consequently, $\det(L) = +1$. Similarly, LU factorization of $C_4^{\mathrm{IV}}J_4$ given by (5.147) can be converted to DLU scaling up the matrix L by a diagonal matrix D. Then L will have unit-diagonal elements.

As the matrix C_4^{II} in (5.135) can be again factorized according to the EOT factorization defined by (5.136) and (5.137), where

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}, \quad C_2^{\mathrm{IV}}J_2 = \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix}, \tag{5.148}$$

we can replace LU factorization of C_4^{II} given by (5.146) and by the EOT factorization of $\hat{C}_4^{\mathrm{II}}$ with substituted LU factorizations of matrices C_2^{II} and $C_2^{\mathrm{IV}}J_2$. According to (5.144) the LU matrix product is given by

$$C_2^{\mathrm{II}} = C_2^{\mathrm{IV}}J_2 = LU = \begin{pmatrix} l_{11} & l_{11}u_{12} \\ l_{21} & l_{21}u_{12} + l_{22} \end{pmatrix}. \tag{5.149}$$

Again, comparing the corresponding matrix elements in (5.149) and solving the system of linear equations for matrices C_2^{II} and $C_2^{\mathrm{IV}}J_2$ we obtain

$$C_2^{\mathrm{II}} = LU = \begin{pmatrix} \frac{\sqrt{2}}{2} & 0 \\ \frac{\sqrt{2}}{2} & -\sqrt{2} \end{pmatrix}\begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix},$$
$$C_2^{\mathrm{IV}}J_2 = LU = \begin{pmatrix} \cos\frac{3\pi}{8} & 0 \\ -\sin\frac{3\pi}{8} & \frac{1}{\cos\frac{3\pi}{8}} \end{pmatrix}\begin{pmatrix} 1 & \tan\frac{3\pi}{8} \\ 0 & 1 \end{pmatrix}. \tag{5.150}$$

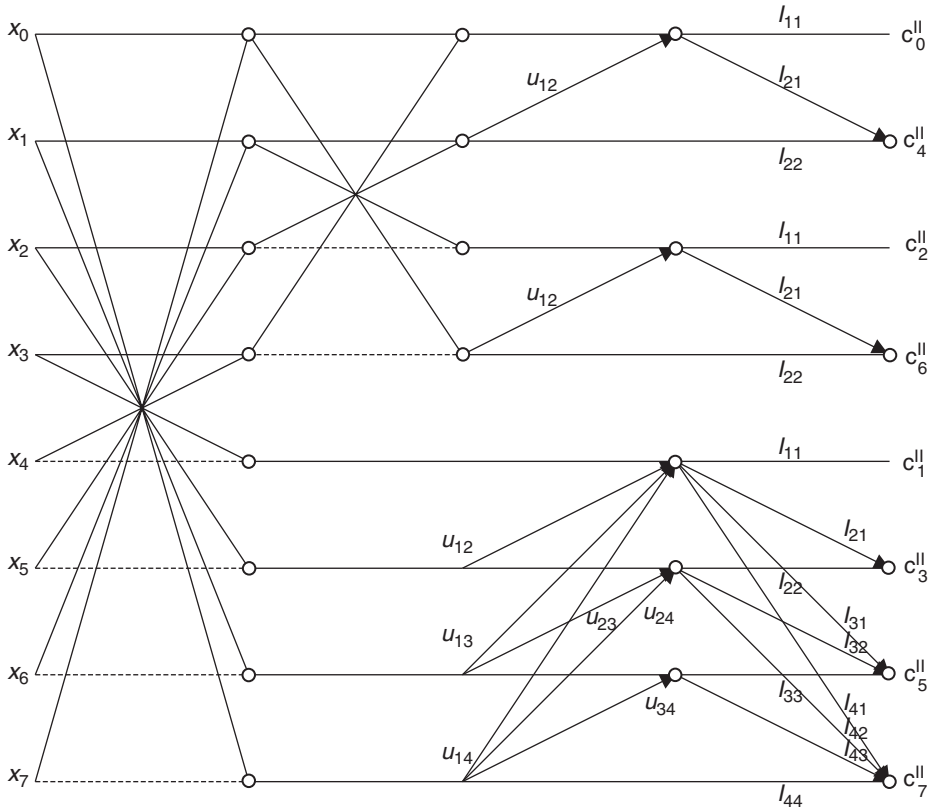


Fig. 5.31. The *LU*-based structure for the 8-point DCT-II computation.

Substituting the analytically derived *LU* factorizations of matrices C_2^{II}, $C_2^{\mathrm{IV}} J_2$ and $C_4^{\mathrm{IV}} J_4$ into the modified EOT factorizations of $\hat{C}_8^{\mathrm{II}}$ and $\hat{C}_4^{\mathrm{II}}$ defined by (5.135) and (5.136), respectively, we obtain the regular *LU*-based structure for the forward 8-point DCT-II computation which is shown in Fig. 5.31. The normalization factors for output DCT-II coefficients are defined by $\hat{D}_8 = \mathrm{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$. Inverses of L, U matrices exist and are triangular. Therefore, the computational structure for inverse 8-point DCT-II computation, DCT-III, is simply obtained by reversing each stage of the forward *LU*-based structure. Introducing the rounding operator we obtain the 8-point *LU*-based LDCT-II structure.

5.5.1.4 PLUS-based factorization of DCT matrices

PLUS factorization of a real square nonsingular matrix A of order N with $\det(A) = +1$ is stated by Theorem 5.3 in Section 5.2.7, where P is a permutation matrix, L is a unit lower triangular, U is a unit upper triangular and S is a unit lower triangular matrix. If we apply the *PLUS* factorization algorithm described in Section 5.2.7 to the eigenorthonormal

matrices C_4^{II}, $C_4^{\mathrm{IV}} J_4$ given by (5.133) we respectively obtain

$$C_4^{\mathrm{II}} = PLUS$$

$$= \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & -1 & 1 & 0 \\ 0 & \left(\tan\frac{\pi}{8} - \frac{1}{\sqrt{2}\cos\frac{\pi}{8}}\right) & \frac{3}{\sqrt{2}\cos\frac{\pi}{8}} & 1 \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & \frac{1}{2} \\ 0 & 1 & 2 & -\frac{\sqrt{2}}{2}\cos\frac{\pi}{8} \\ 0 & 0 & 1 & -\frac{\sqrt{2}}{2}\cos\frac{\pi}{8} \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ -1 & \left(\frac{\sqrt{2}}{\cos\frac{\pi}{8}} - \tan\frac{\pi}{8}\right) & \left(\frac{2\sqrt{2}}{\cos\frac{\pi}{8}} + \tan\frac{\pi}{8}\right) & 1 \end{pmatrix}, \quad P = I_4, \tag{5.151}$$

where

$$u_{12} = \frac{1}{2}\left(1 - \frac{\sqrt{2}}{\cos\frac{\pi}{8}} + \tan\frac{\pi}{8}\right),$$

$$u_{13} = \frac{1}{2}\left(1 - \frac{\sqrt{2}}{\cos\frac{\pi}{8}} - \tan\frac{\pi}{8}\right),$$

and similarly,

$$C_4^{\mathrm{IV}} J_4 = PLUS = \begin{pmatrix} 1 & 0 & 0 & 0 \\ l_{21} & 1 & 0 & 0 \\ l_{31} & l_{32} & 1 & 0 \\ l_{41} & l_{42} & l_{42} & 1 \end{pmatrix} \begin{pmatrix} 1 & u_{12} & u_{13} & \frac{\sqrt{2}}{2}\cos\frac{\pi}{16} \\ 0 & 1 & u_{23} & \frac{\sqrt{2}}{4} \\ 0 & 0 & 1 & \frac{1}{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\times \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ s_1 & s_2 & s_3 & 1 \end{pmatrix}, \quad P = I_4, \tag{5.152}$$

where

$$l_{21} = \frac{\sqrt{2}}{2} - \frac{1}{2\cos\frac{\pi}{16}} + \frac{\sqrt{2}}{2}\tan\frac{\pi}{16},$$

$$l_{31} = \frac{\sqrt{2}}{2} + \frac{1}{2\cos\frac{\pi}{16}} - \frac{\sqrt{2}}{2}\tan\frac{\pi}{16},$$

$$l_{41} = \frac{-\frac{\sqrt{2}}{2} + \cos\frac{\pi}{16} + \sin\frac{\pi}{16} - \frac{1}{\cos\frac{\pi}{16}} + \frac{\sqrt{2}}{2}\tan\frac{\pi}{16}}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}},$$

$$l_{32} = -1 + \sqrt{2}\left(\sin\frac{\pi}{16} - \cos\frac{\pi}{16}\right),$$

$$l_{42} = \frac{\sqrt{2}(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}) - 2\cos\frac{\pi}{16} - \sin\frac{\pi}{16} + 2\cos^2\frac{\pi}{16}}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}},$$

$$l_{43} = \frac{-\sqrt{2}\cos\frac{\pi}{16} - 2}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}},$$

$$u_{12} = 2\cos\frac{\pi}{16} - \frac{\sqrt{2}}{2} + 2\sqrt{2}\cos^2\frac{\pi}{16},$$

$$u_{13} = \frac{\frac{3\sqrt{2}}{2} + 2\cos^2\frac{\pi}{16} - \frac{\sqrt{2}}{2}\sin\frac{\pi}{16} + 2\cos\frac{\pi}{16}\sin\frac{\pi}{16}(1 + \sqrt{2}\cos\frac{\pi}{16} - \sqrt{2}\sin\frac{\pi}{16})}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}},$$

$$u_{23} = \frac{\frac{\sqrt{2}}{2} + \cos\frac{\pi}{16} + \sin\frac{\pi}{16}}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}},$$

and

$$s_1 = \tan\frac{\pi}{16} - \frac{\sqrt{2}}{\cos\frac{\pi}{16}},$$

$$s_2 = -\frac{3\sqrt{2}}{2} + \frac{1}{\cos\frac{\pi}{16}} - \frac{\sqrt{2}}{2}\tan\frac{\pi}{16} - 4\cos\frac{\pi}{16},$$

$$s_3 = \frac{-3 - \sqrt{2}\cos\frac{\pi}{16} - \frac{\sqrt{2}}{2\cos\frac{\pi}{16}} + \tan\frac{\pi}{16} - 4\cos\frac{\pi}{16}\sin\frac{\pi}{16} - 2\sqrt{2}\sin\frac{\pi}{16} + 4\sin^2\frac{\pi}{16}}{\cos\frac{\pi}{16} - \sin\frac{\pi}{16}}.$$

The matrix C_4^{II} in (5.135) can again be factorized according to the EOT factorization defined by (5.136), where $C_2^{\mathrm{II}} = G_{-\frac{\pi}{4}} J_2$ and $C_2^{\mathrm{IV}} J_2 = G_{-\frac{3\pi}{8}}$ are Givens–Jacobi rotations

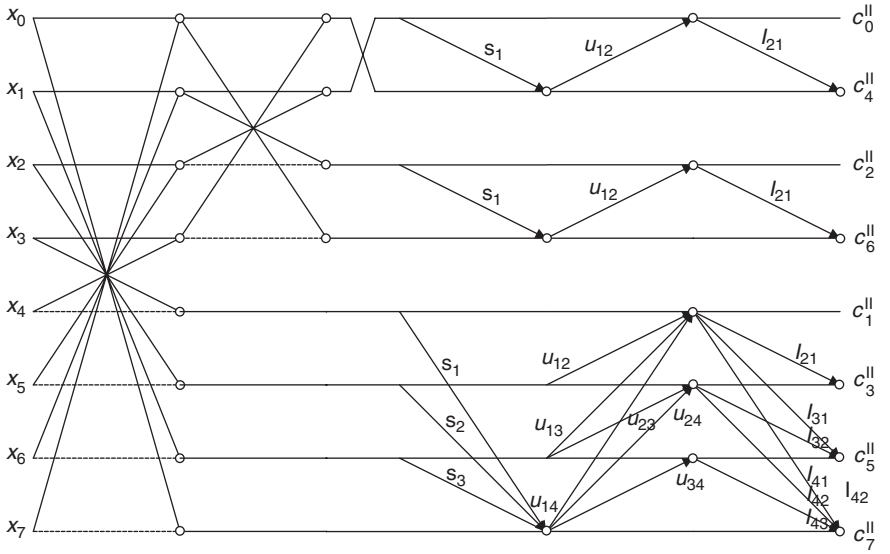


Fig. 5.32. The *PLUS*-based structure for the 8-point DCT-II computation.

which have also *PLUS* factorizations as

$$C_2^{\mathrm{II}} = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ -\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{pmatrix} J_2 = \begin{pmatrix} 1 & 0 \\ 1-\sqrt{2} & 1 \end{pmatrix} \begin{pmatrix} 1 & \frac{\sqrt{2}}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 1-\sqrt{2} & 1 \end{pmatrix} J_2,$$

$$C_2^{\mathrm{IV}} J_2 = \begin{pmatrix} \cos\frac{3\pi}{8} & \sin\frac{3\pi}{8} \\ -\sin\frac{3\pi}{8} & \cos\frac{3\pi}{8} \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ -\tan\frac{3\pi}{8} & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\frac{3\pi}{8} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\tan\frac{3\pi}{8} & 1 \end{pmatrix}. \tag{5.153}$$

In fact, the LUL structures of G_φ correspond to *PLUS* factorizations. Substituting the analytically derived *PLUS* factorizations of matrices C_2^{II}, $C_2^{\mathrm{IV}} J_2$ and $C_4^{\mathrm{IV}} J_4$ into the modified EOT factorizations of $\hat{C}_8^{\mathrm{II}}$ and $\hat{C}_4^{\mathrm{II}}$ defined by (5.135) and (5.136), respectively, we obtain the regular *PLUS*-based structure for the forward 8-point DCT-II computation which is shown in Fig. 5.32. The normalization factors for output DCT-II coefficients are defined by $\hat{D}_8 = \mathrm{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$. Inverses of L, U and S matrices exist and are triangular. Therefore, the computational structure for inverse 8-point DCT-II computation, DCT-III, is obtained by reversing each stage of the forward *PLUS*-based structure. Introducing the rounding operator we obtain the *PLUS*-based 8-point LDCT-II structure.

5.5.1.5 Optimization of LU- and PLUS-based computational structures

Every linear transform built with *LU* or *PLUS* factorization is immediately integer reversible and its inverse has exactly the same computational complexity as the forward

transform. *LU* and *PLUS* factorizations define the regular computational structures which can be implemented in-place. The factorization of a specific DCT/DST matrix may be practically optimized for lower computational complexity and less error by searching for a proper permutation matrix. An optimization scheme can be considered as a minimization of [9]:

- the number of factored matrices,
- the computational complexity of each individual step,
- the error with respect to the theoretical computation.

However, such optimization is not easy in practice. The basic method involves complete pivoting, i.e., applying row and column permutations to the transform matrix. Chosen pivots bring forth the most zeros in factored matrices and the most equal elements in a row.

5.5.1.6 Error estimation method for LU and PLUS factorizations

Since rounding operations are applied to all noninteger elements in the factored matrices, truncation errors are inevitable. The rounding-off errors are in the interval $\langle -0.5, 0.5\rangle$. In order to estimate errors we use matrix and vector norms (maximum or ∞-norm) and scalar u to denote the unit round-off error which is defined as the largest error that can occur in a rounding operation, i.e., 0.5. Let $\mathbf{u_m}$ denotes the rounding error vector resulting from the transformation of the mth triangular matrix. If all elements in the mth factored matrix are integers then, $\mathbf{u_m} = 0$, and the matrix does not produce any errors, but it still transfers and propagates existent errors produced by previous matrices.

For the *PLUS* factorization, the total rounding error is defined as [9]

$$\mathbf{u} = P(\mathbf{u_1} + L(\mathbf{u_2} + U\mathbf{u_3})) = P(\mathbf{u_1} + L\mathbf{u_2} + LU\mathbf{u_3}), \tag{5.154}$$

where $\mathbf{u_1}$, $\mathbf{u_2}$ and $\mathbf{u_3}$ are error vectors for matrices L, U and S, respectively, and only the Nth element of $\mathbf{u_3}$ is not 0. Hence,

$$|\mathbf{u}| = |P(\mathbf{u_1} + L\mathbf{u_2} + LU\mathbf{u_3})| \leq (|\mathbf{u_1}| + |L\mathbf{u_2}| + |LU\mathbf{u_3}|), \tag{5.155}$$

where $|\cdot|$ denotes absolute values of all elements in a vector. An error bound can be estimated as [9]

$$\|\mathbf{u}\|_\infty \leq u \cdot (1 + \|L\|_\infty + \|LU\mathbf{e_N}\|_\infty), \tag{5.156}$$

where $\mathbf{e_N} = [0, 0, \ldots, 0, 1]$. Equation (5.155) implies that the total rounding error and error bound for *LU* factorization are respectively defined as

$$|\mathbf{u}| \leq (|\mathbf{u_1}| + |L\mathbf{u_2}|) \tag{5.157}$$

and

$$\|\mathbf{u}\|_\infty \leq u \cdot (1 + \|L\|_\infty). \tag{5.158}$$

By numerical evaluation of *LU* factorizations of the transform matrix $C_4^{\mathrm{IV}} J_4$ given by (5.147), and transform matrices C_2^{II}, $C_2^{\mathrm{IV}} J_2$ given by (5.150) which define the 8-point *LU*-based LDCT-II structure shown in Fig. 5.31, the total rounding error and error bound are respectively estimated as

$$|\mathbf{u}| \leq |[1, 1, 1.38268, 1.92388, 1.13795, 1.03237, 2.40333, 5.82782]^{\mathrm{T}}|,$$

and

$$\|\mathbf{u}\|_\infty \leq u \cdot 18.20268.$$

Similarly, by numerical evaluation of *PLUS* factorizations of the transform matrix $C_4^{\mathrm{IV}} J_4$ given by (5.152), and transform matrices C_2^{II}, $C_2^{\mathrm{IV}} J_2$ given by (5.153) which define the 8-point *PLUS*-based LDCT-II structure shown in Fig. 5.32, the total rounding error and error bound are respectively estimated as

$$|\mathbf{u}| \leq |[1.70711, 2.12132, 1.92388, 2.05086, 1.69352, 2.92563, 2.42774, 2.59510]^{\mathrm{T}}|,$$

and

$$\|\mathbf{u}\|_\infty \leq u \cdot 10.90223.$$

QR-, *LU*- and *PLUS*-based factorizations of DCT matrices provide the regular computational structures and introducing rounding operators the corresponding *QR*-, *LU*- and *PLUS*-based LDCTs with reversible integer-to-integer mapping are obtained.

5.5.2 Invertible integer DCTs

Integer approximated DCT-II and DCT-IV, called invertible integer DCTs, have been proposed in Refs. [59–61]. The approach to construction of invertible integer DCTs is based on recursive factorizations of DCT-II and DCT-IV matrices into products of sparse orthogonal matrices of simple structure. The derived invertible integer DCTs are very close to the original DCTs and map integer vectors to integer vectors. By suitable permutations, each matrix factor is transformed into block-diagonal form where each block is an orthogonal matrix of order 2, and thus the construction of invertible integer DCTs is reduced to the construction of only integer transforms of length 2. Each 2×2 orthogonal (rotation) matrix is factorized into a product of Gauss–Jordan elementary matrices defining the LUL or equivalently ULU factorization (see Section 5.2.7), and the rounding procedure in corresponding LUL or ULU structure is added to obtain integer-to-integer transform whereby the truncation errors are explicitly estimated. In particular, for the fast integer DCT algorithms explicit error bounds for the difference between results of exact DCT and the related integer DCT in the Euclidean and maximum norm are estimated [59].

In general, the method for construction of invertible integer DCTs is quite similar to that of BinDCT discussed in Section 5.4.4 where the multipliers in LUL or ULU structure are approximated by dyadic rationals. Although the DCT-II and DCT-IV are linear mappings which are generated by the corresponding matrices, naturally, by introducing a nonlinear rounding operation into computational steps, the invertible integer DCTs are

no longer linear mappings. Thus, the invertible integer DCT is understood to be nonlinear, (left)-invertible mapping which acts on Z^N (N-dimensional vector space over Z) and approximates the classical DCT. The results can be directly extended to the 2-D integer DCT. Using the (recursive) factorization of a DCT/DST matrix, the method can easily be applied to any other trigonometric transform.

5.5.2.1 Orthogonal factorizations of DCT-II and DCT-IV matrices

Let $N \geq 4$ be an even integer, and let C_N^{II} and C_N^{IV} be the DCT-II and DCT-IV matrices of order N defined by (4.2) and (4.4), respectively. Then, the DCT-II matrix C_N^{II} can be recursively factorized into the product of sparse orthogonal matrices as [59, 60]

$$C_N^{\mathrm{II}} = P_N^{\mathrm{T}} \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{IV}} \end{pmatrix} T_N^{(0)}, \tag{5.159}$$

with the orthogonal matrix

$$T_N^{(0)} = \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ I_{\frac{N}{2}} & -J_{\frac{N}{2}} \end{pmatrix} = \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & J_{\frac{N}{2}} \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_{\frac{N}{2}} & J_{\frac{N}{2}} \\ J_{\frac{N}{2}} & -I_{\frac{N}{2}} \end{pmatrix}.$$

The DCT-IV matrix C_N^{IV} can be recursively factorized into the product of sparse orthogonal matrices as [59, 60]

$$C_N^{\mathrm{IV}} = P_N^{\mathrm{T}} A_N \begin{pmatrix} C_{\frac{N}{2}}^{\mathrm{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\mathrm{II}} \end{pmatrix} T_N^{(1)}, \tag{5.160}$$

with the orthogonal matrices

$$A_N = \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & & & 0 \\ & I_{\frac{N}{2}-1} & I_{\frac{N}{2}-1} & \\ & I_{\frac{N}{2}-1} & -I_{\frac{N}{2}-1} & \\ 0 & & & -\sqrt{2} \end{pmatrix} \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & D_{\frac{N}{2}} J_{\frac{N}{2}} \end{pmatrix},$$

$$T_N^{(1)} = \begin{pmatrix} I_{\frac{N}{2}} & 0 \\ 0 & D_{\frac{N}{2}} \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{4N} & & & & & \sin\frac{\pi}{4N} \\ & \cos\frac{3\pi}{4N} & & & \sin\frac{3\pi}{4N} & \\ & & \ddots & \iddots & & \\ & & \cos\frac{(N-1)\pi}{4N} & \sin\frac{(N-1)\pi}{4N} & & \\ & & -\sin\frac{(N-1)\pi}{4N} & \cos\frac{(N-1)\pi}{4N} & & \\ & & \iddots & \ddots & & \\ & -\sin\frac{3\pi}{4N} & & & \cos\frac{3\pi}{4N} & \\ -\sin\frac{\pi}{4N} & & & & & \cos\frac{\pi}{4N} \end{pmatrix},$$

where $D_{\frac{N}{2}} = \text{diag}\{(-1)^k\}$, $k = 0, 1, \ldots, \frac{N}{2} - 1$, is the diagonal odd-sign changing matrix, and P_N is the even–odd permutation matrix given by $P_N\mathbf{x} = [x_0, x_2, \ldots, x_{N-2}, x_1, x_3, \ldots, x_{N-1}]^{\mathrm{T}}$, where $P_N^{-1} = P_N^{\mathrm{T}}$. From (5.159) and (5.160) it can be easily seen that up to permutations and changes of sign, the orthogonal matrices $T_N^{(0)}$, A_N and $T_N^{(1)}$ can be represented as block-diagonal matrices $R_2(\varphi)$ of order 2 of the form:

$$R_2\left(\frac{\pi}{4}\right) = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ \sin\frac{\pi}{4} & -\cos\frac{\pi}{4} \end{pmatrix} = H_{\frac{\pi}{4}},$$

being Householder reflections or

$$R_2(\varphi) = \begin{pmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{pmatrix} = G_{-\varphi}, \quad \varphi = \frac{(2k+1)\pi}{4N}, \quad k = 0, 1, \ldots, \frac{N}{2} - 1,$$

being the Givens–Jacobi rotation matrices. The constructions of invertible integer DCT-II and DCT-IV are based on this essential fact.

5.5.2.2 Integer transforms of the length 2

The main idea to obtain an invertible integer DCT is as follows. For a given invertible matrix R_2 and for arbitrary vector $\mathbf{x} \in Z^2$, find a suitable integer approximation of $R_2\mathbf{x}$ such that this process is invertible. The simple structure of the matrix factors of C_N^{II} and C_N^{IV} implies that it is necessary to find a suitable solution only for the orthogonal matrices $R_2(\varphi)$ (being Householder reflections or Givens–Jacobi rotations) with rotation angles $\varphi \in (0, \frac{\pi}{4})$.

From Section 5.2.7 we know that Givens–Jacobi rotations $G_{-\varphi}$ and G_{φ} (forward and inverse) can be represented as a product of three Gauss–Jordan elementary matrices that are unit lower and unit upper triangular matrices defining LUL factorization (see (5.20) and (5.21)) or equivalently ULU factorization (see (5.22) and (5.23)). Here we prefer the use of forward and inverse ULU factorizations of $G_{-\varphi}$ and G_{φ}, respectively, which are defined as

$$\begin{aligned} G_{-\varphi} &= \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}, \\ G_{\varphi} &= \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\varphi & 1 \end{pmatrix} \begin{pmatrix} 1 & -\tan\frac{\varphi}{2} \\ 0 & 1 \end{pmatrix}, \quad \sin\varphi \neq 0. \end{aligned} \tag{5.161}$$

The corresponding ULU structures are shown in Fig. 5.2(a) and (b). The ULU factorizations consist of nonorthogonal matrix factors which are still invertible. These factorizations can be used for the construction of invertible integer DCT as follows.

For $a \in R$ let $\lfloor a \rfloor = \max\{x \leq a : x \in Z\}$ and $\{a\} = a - \lfloor a \rfloor \in \langle 0, 1)$. Then $\{a\}$ is the noninteger part of a. Further, let $\text{round}\,(a) = \lfloor a + \frac{1}{2} \rfloor$ be the integer closest to a. Consider the last matrix

factor in the right-hand side of (5.161). One computational step in the matrix–vector form can be written as

$$\hat{\mathbf{y}} = \begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \mathbf{x}, \quad s \in R,$$

where $\mathbf{x} = [x_0, x_1]^{\mathrm{T}} \in Z^2$. Then, $\hat{\mathbf{y}} = [\hat{y}_0, \hat{y}_1]^{\mathrm{T}}$ can be approximated by $\mathbf{y} = [y_0, y_1]^{\mathrm{T}} \in Z^2$ with

$$y_0 = x_0 + \left\lfloor sx_1 + \frac{1}{2} \right\rfloor = x_0 + \mathrm{round}(sx_1), \quad y_1 = x_1.$$

This transform is invertible with

$$x_0 = y_0 - \left\lfloor sy_1 + \frac{1}{2} \right\rfloor = y_0 - \mathrm{round}(sy_1), \quad x_1 = y_1.$$

Indeed, we have

$$y_0 - \mathrm{round}(sy_1) = x_0 + \mathrm{round}(sx_1) - \mathrm{round}(sx_1) = x_0, \quad y_1 = x_1.$$

The following theorem is fundamental for the integer approximation of ULU factorized rotation matrix $R_2 = R_2(\varphi)$ by rounding procedure with explicit estimates of truncation errors [59].

Theorem 5.4: (Rounding procedure of the rotation matrix $R_2(\varphi)$ represented by ULU structure with explicit truncation error estimates)

Let $R_2 = R_2(\varphi)$ with $\varphi \in (0, \frac{\pi}{4})$ be a rotation matrix represented by ULU structure. Then for arbitrary $\mathbf{x} = [x_0, x_1]^{\mathrm{T}} \in Z^2$, a suitable integer approximation $\mathbf{y} = [y_0, y_1]^{\mathrm{T}} \in Z^2$ of $\hat{\mathbf{y}} = R_2\mathbf{x}$ is given by $y_0 = z_2$, $y_1 = z_1$, where

$$z_0 = x_0 + \mathrm{round}\left(x_1 \tan\frac{\varphi}{2}\right), \quad z_1 = x_1 + \mathrm{round}(-z_0 \sin\varphi), \quad z_2 = z_0 + \mathrm{round}\left(z_1 \tan\frac{\varphi}{2}\right).$$

The procedure is invertible and its inverse is $x_0 = v_2$, $x_1 = v_1$, where

$$v_0 = y_0 - \mathrm{round}\left(y_1 \tan\frac{\varphi}{2}\right), \quad v_1 = y_1 - \mathrm{round}(-v_0 \sin\varphi), \quad v_2 = v_0 - \mathrm{round}\left(v_1 \tan\frac{\varphi}{2}\right).$$

Further, the truncation error can be estimated by

$$\|\hat{\mathbf{y}} - \mathbf{y}\|_2 \le \sqrt{h(\varphi)}, \quad \|\hat{\mathbf{y}} - \mathbf{y}\|_\infty \le g(\varphi), \tag{5.162}$$

where functions $h(\varphi)$ and $g(\varphi)$ are respectively given by

$$h(\varphi) = \frac{3}{4} + \sin\varphi + \frac{1}{2}\cos\varphi + \frac{1}{4}\tan^2\frac{\varphi}{2}, \quad g(\varphi) = \frac{1}{2}\left(1 + \tan\frac{\varphi}{2} + \cos\varphi\right).$$

•

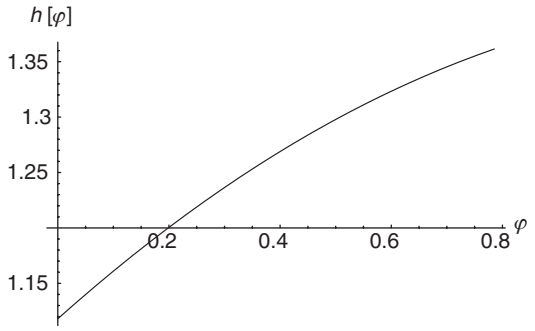


Fig. 5.33. Plot of the function $h(\varphi)$.

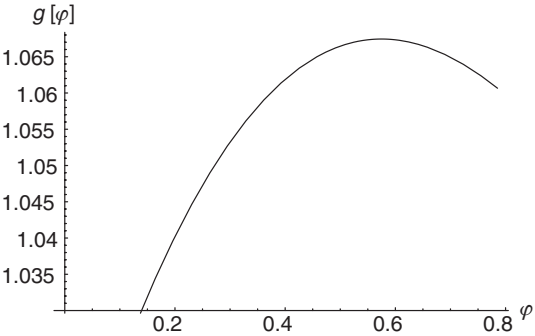


Fig. 5.34. Plot of the function $g(\varphi)$.

The formulae for y_0, y_1 and x_0, x_1 (after inverse transform) directly follow by applying the input vector $\mathbf{x}$ to three matrices in (5.161). The elegant proof of truncation error estimates given by (5.162) can be found in Ref. [59]. Plots of the functions $h(\varphi)$ and $g(\varphi)$ are respectively shown in Figs. 5.33 and 5.34.

Note 1: Since there exists the relation between Givens–Jacobi rotation and Householder reflection given by (5.36) and (5.37), the procedure of Theorem 5.4 can also be obtained for Householder reflections H_φ. In this case, the integer approximation $\mathbf{y} = [y_0, y_1]^{\mathrm{T}} \in Z^2$ of $R_2\mathbf{x}$ is of the form $y_0 = z_2$, $y_1 = -z_1$ with z_0, z_1, z_2 as in Theorem 5.4, and the truncation error estimates hold as before.

Note 2: Let $\hat{\mathbf{y}} = R_2(\varphi)\mathbf{x}$ with arbitrary vector $\mathbf{x} \in Z^2$ be given and let $\mathbf{y}$ be its integer approximation. The special values for truncation errors $\|\hat{\mathbf{y}} - \mathbf{y}\|_2$ and $\|\hat{\mathbf{y}} - \mathbf{y}\|_\infty$ via the rounding procedure for rotation angles $\varphi \in \left\{\frac{\pi}{4}, \frac{\pi}{8}, \frac{\pi}{16}, \frac{3\pi}{16}\right\}$ can be obtained by substituting values of φ into formulae (5.162). In particular, we obtain

$$\|\hat{\mathbf{y}} - \mathbf{y}\|_2 \leq \begin{cases} 1.361453 & \text{for } \varphi = \frac{\pi}{4}, \\ 1.266694 & \text{for } \varphi = \frac{\pi}{8}, \\ 1.199128 & \text{for } \varphi = \frac{\pi}{16}, \\ 1.320723 & \text{for } \varphi = \frac{3\pi}{16}, \end{cases} \qquad \|\hat{\mathbf{y}} - \mathbf{y}\|_\infty \leq \begin{cases} 1.060660 & \text{for } \varphi = \frac{\pi}{4}, \\ 1.061396 & \text{for } \varphi = \frac{\pi}{8}, \\ 1.039638 & \text{for } \varphi = \frac{\pi}{16}, \\ 1.067408 & \text{for } \varphi = \frac{3\pi}{16}. \end{cases}$$

Further, for all $\varphi \in \langle 0, \frac{\pi}{4})$ we have

$$\|\hat{\mathbf{y}} - \mathbf{y}\|_\infty \le \max\left\{g(\varphi)\colon \varphi \in \left\langle 0, \frac{\pi}{4}\right)\right\} \approx 1.067442.$$

5.5.2.3 Fast invertible integer DCT algorithms

The method to integer approximation of $R_2(\varphi)$ described by Theorem 5.4 enables us to derive the fast invertible integer DCT-II and integer DCT-IV algorithms of the length $N = 2^m$ based on recursive matrix factorizations of C_N^{II} and C_N^{IV} defined by (5.159) and (5.160), respectively.

Algorithm A: The first step is to apply the rounding procedure to all rotation matrices $R_2(\varphi)$ represented by ULU structure in the orthogonal matrix factors of C_N^{II} and C_N^{IV}. In this way we get a direct integer approximation of $C_N^{\mathrm{II}}\mathbf{x}$ and $C_N^{\mathrm{IV}}\mathbf{x}$. The inverse integer DCT computed by algorithm A is obtained by going backwards and taking the inverse procedures defined by Theorem 5.4. It is noted that integer DCTs realized by algorithm A are invertible on Z^N.

Consider the 8-point DCT-II computation. For $N = 8$, combining the sparse matrix factorizations of C_N^{II} and C_N^{IV} matrices defined by (5.159) and (5.160), respectively, we obtain the following orthogonal factorization of C_8^{II} transform matrix as [59]

$$\begin{aligned} C_8^{\mathrm{II}} &= P_8^{\mathrm{T}} \begin{pmatrix} C_4^{\mathrm{II}} & 0 \\ 0 & C_4^{\mathrm{IV}} \end{pmatrix} T_8^{(0)} \\ &= B_8 \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \begin{pmatrix} C_2^{\mathrm{II}} & & & 0 \\ & C_2^{\mathrm{IV}} & & \\ & & C_2^{\mathrm{II}} & \\ 0 & & & C_2^{\mathrm{II}} \end{pmatrix} \begin{pmatrix} T_4^{(0)} & 0 \\ 0 & T_4^{(1)} \end{pmatrix} T_8^{(0)}, \end{aligned} \tag{5.163}$$

where B_8 is the bit-reversal matrix, and matrices A_4, C_2^{II}, C_2^{IV}, $T_4^{(0)}$, $T_8^{(0)}$ and $T_4^{(1)}$ are respectively given by

$$A_4 = \frac{\sqrt{2}}{2} \begin{pmatrix} \sqrt{2} & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & \sqrt{2} & 0 \end{pmatrix},$$

$$C_2^{\mathrm{II}} = \frac{\sqrt{2}}{2} \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} = H_{\frac{\pi}{4}},$$

$$C_2^{IV} = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} I_2 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ -\sin\frac{\pi}{8} & \cos\frac{\pi}{8} \end{pmatrix} = \begin{pmatrix} I_2 & 0 \\ 0 & D_2 \end{pmatrix} G_{-\frac{\pi}{8}},$$

$$T_4^{(0)} = \frac{\sqrt{2}}{2} \begin{pmatrix} I_2 & J_2 \\ I_2 & -J_2 \end{pmatrix} = \begin{pmatrix} I_2 & 0 \\ 0 & J_2 \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix},$$

$$T_8^{(0)} = \frac{\sqrt{2}}{2} \begin{pmatrix} I_4 & J_4 \\ I_4 & -J_4 \end{pmatrix} = \begin{pmatrix} I_4 & 0 \\ 0 & J_4 \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix},$$

$$T_4^{(1)} = \begin{pmatrix} I_2 & 0 \\ 0 & D_2 \end{pmatrix} \begin{pmatrix} \cos\frac{\pi}{16} & 0 & 0 & \sin\frac{\pi}{16} \\ 0 & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & 0 \\ 0 & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & 0 \\ -\sin\frac{\pi}{16} & 0 & 0 & \cos\frac{\pi}{16} \end{pmatrix},$$

where $D_2 = \text{diag}\{1, -1\}$. The orthogonal factorization (5.163) defines the fast 8-point DCT-II algorithm involving a total of 13 rotations. Note that this factorization with scaling $\sqrt{2}$ implies a fast 8-point scaled DCT-II algorithm with 11 multiplications and 29 additions similar to that of Ref. [20]. Applying the rounding procedure to all rotation matrices we get the fast 8-point integer DCT-II algorithm with 39 rounding operations. Due to the large number of rotations in matrix factors the arithmetic complexity of such an algorithm is relatively high. However, admitting a scaling factor an alternative integer DCT-II with smaller arithmetic complexity can be obtained.

Algorithm B: By introducing the scaling factors $\sqrt{N_1}$ and $\sqrt{N_2}$ for the C_N^{II} and C_N^{IV} matrices, where $N_1 = \frac{N}{2}$ and $N_2 = \frac{N}{4}$ the following factorizations for $N \geq 8$ are proposed [59]:

$$\sqrt{N_1}\, C_N^{\text{II}} = P_N^{\text{T}} \sqrt{N_2} \begin{pmatrix} C_{\frac{N}{2}}^{\text{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\text{IV}} \end{pmatrix} \sqrt{2}\, T_N^{(0)}, \tag{5.164}$$

$$\sqrt{N_1}\, C_N^{\text{IV}} = P_N^{\text{T}} \sqrt{2} A_N \sqrt{N_2} \begin{pmatrix} C_{\frac{N}{2}}^{\text{II}} & 0 \\ 0 & C_{\frac{N}{2}}^{\text{II}} \end{pmatrix} T_N^{(1)}, \tag{5.165}$$

starting with

$$\sqrt{2} C_4^{\text{II}} = P_4^{\text{T}} \begin{pmatrix} C_2^{\text{II}} & 0 \\ 0 & C_2^{\text{IV}} \end{pmatrix} \sqrt{2}\, T_4^{(0)},$$

$$\sqrt{2} C_4^{\text{IV}} = P_4^{\text{T}} A_4 \sqrt{2} \begin{pmatrix} C_2^{\text{II}} & 0 \\ 0 & C_2^{\text{II}} \end{pmatrix} T_4^{(1)},$$

where in the factorization of $\sqrt{2}C_4^{\mathrm{IV}}$ the scaling factor $\sqrt{2}$ is used for the matrix factor

$$\begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & C_2^{\mathrm{II}} \end{pmatrix}$$

differing from the rule for $N \geq 8$. The matrix factor $\sqrt{2}C_2^{\mathrm{II}}$ generates a left-invertible mapping on Z^2 and, therefore, integer DCTs realized by algorithm B are only left invertible on Z^N.

For $N = 8$, combining the sparse matrix factorizations of scaled C_N^{II} and C_N^{IV} defined by (5.164) and (5.165), respectively, we obtain the following factorization of scaled $2C_8^{\mathrm{II}}$ matrix as [59]

$$2C_8^{\mathrm{II}} = B_8 \begin{pmatrix} I_4 & 0 \\ 0 & A_4 \end{pmatrix} \begin{pmatrix} C_2^{\mathrm{II}} & & & 0 \\ & C_2^{\mathrm{IV}} & & \\ & & \sqrt{2}C_2^{\mathrm{II}} & \\ 0 & & & \sqrt{2}C_2^{\mathrm{II}} \end{pmatrix} \begin{pmatrix} \sqrt{2}T_4^{(0)} & 0 \\ 0 & T_4^{(1)} \end{pmatrix} \sqrt{2}T_8^{(0)}. \tag{5.166}$$

The factorization (5.166) defines the fast 8-point scaled DCT-II algorithm. The corresponding signal flow graph for the fast 8-point scaled DCT-II with highlighted Householder reflections and Givens–Jacobi rotations is shown in Fig. 5.35. The matrices $\sqrt{2}C_2^{\mathrm{II}}$, $\sqrt{2}T_4^{(0)}$ and $\sqrt{2}T_8^{(0)}$ contain only integers and rounding procedures are not necessary and the number of rotations is reduced to only 5. Applying the rounding procedure to all rotation matrices we get the fast 8-point integer scaled DCT-II algorithm with 15 rounding operations.

5.5.2.4 Error bounds between the exact (scaled) DCT-II and corresponding integer (scaled) DCT-II

The rounding procedure defined by Theorem 5.4 enables us to estimate the difference between the results of exact (scaled) DCT-II and the corresponding integer (scaled) DCT-II for both the algorithms A and B.

Let $N = 2^m$, $m \geq 1$, and let $\mathbf{x} \in Z^N$ be an arbitrary input vector. Further, let $\mathbf{y} \in Z^N$ be the resulting integer approximation of $\hat{\mathbf{y}} = C_N^{\mathrm{II}}\mathbf{x}$ obtained by the algorithm A. Denote $e_{N,2}^{\mathrm{II}}$ and $e_{N,\infty}^{\mathrm{II}}$ to be the error bound in the Euclidean norm and maximum norm, respectively, i.e.,

$$e_{N,2}^{\mathrm{II}} = \sup\{\|C_N^{\mathrm{II}}\mathbf{x} - \mathbf{y}\|_2 \colon \mathbf{x} \in Z^N\}, \quad e_{N,\infty}^{\mathrm{II}} = \sup\{\|C_N^{\mathrm{II}}\mathbf{x} - \mathbf{y}\|_\infty \colon \mathbf{x} \in Z^N\}.$$

Analogously, for the integer approximation $\mathbf{w}$ of $C_N^{\mathrm{IV}}\mathbf{x}$ via algorithm A, denote the error bounds by $e_{N,2}^{\mathrm{IV}}$ and $e_{N,\infty}^{\mathrm{IV}}$, i.e.,

$$e_{N,2}^{\mathrm{IV}} = \sup\{\|C_N^{\mathrm{IV}}\mathbf{x} - \mathbf{w}\|_2 \colon \mathbf{x} \in Z^N\}, \quad e_{N,\infty}^{\mathrm{IV}} = \sup\{\|C_N^{\mathrm{IV}}\mathbf{x} - \mathbf{w}\|_\infty \colon \mathbf{x} \in Z^N\}.$$

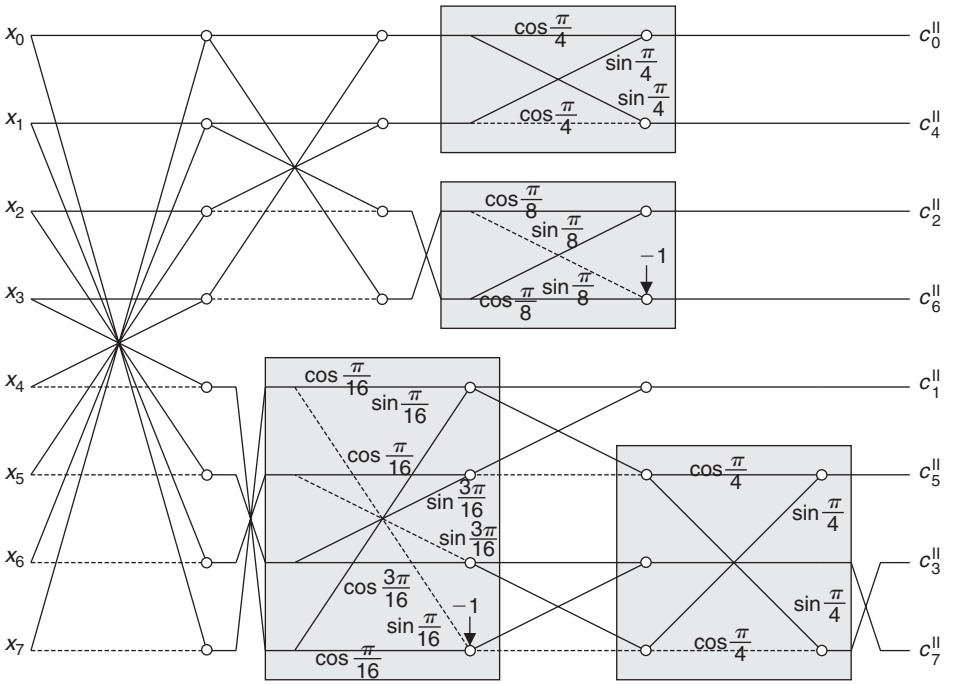


Fig. 5.35. The signal flow graph for the fast 8-point scaled DCT-II with highlighted Householder reflections and Givens–Jacobi rotations.

The error bounds between the results of exact DCT-II and the corresponding integer DCT-II obtained by algorithm A can be estimated by using the following theorem [59].

Theorem 5.5: (Error bounds estimates between the results of exact DCT-II and the corresponding integer DCT-II obtained by algorithm A in the Euclidean and maximum norm)

Let $N = 2^m$, $m \geq 1$, and let $e_{N,2}^{\mathrm{II}}$, $e_{N,\infty}^{\mathrm{II}}$, $e_{N,2}^{\mathrm{IV}}$ and $e_{N,\infty}^{\mathrm{IV}}$ be the error bounds occurring, if exact DCT-II output vectors are compared with the corresponding integer DCT-II results of algorithm A. Then, error bounds in the Euclidean norm can be recursively computed by

$$e_{2,2}^{\mathrm{II}} = \sqrt{h\left(\frac{\pi}{4}\right)}, \quad e_{2,2}^{\mathrm{IV}} = \sqrt{h\left(\frac{\pi}{8}\right)},$$

$$e_{N,2}^{\mathrm{II}} = \sqrt{(e_{N_{1,2}}^{\mathrm{II}})^2 + (e_{N_{1,2}}^{\mathrm{IV}})^2} + \sqrt{N_1\, h\left(\frac{\pi}{4}\right)}, \quad m \geq 2,$$

$$e_{N,2}^{\mathrm{IV}} = \sqrt{\sum_{k=0}^{N_1-1} h\left(\frac{(2k+1)\pi}{4N}\right)} + \sqrt{2}e_{N_{1,2}}^{\mathrm{II}} + \sqrt{(N_1-1)\, h\left(\frac{\pi}{4}\right)}, \quad m \geq 2.$$

Error bounds in maximum norm are

$$e_{2,\infty}^{\mathrm{II}} = g\left(\frac{\pi}{4}\right), \quad e_{2,\infty}^{\mathrm{IV}} = g\left(\frac{\pi}{8}\right),$$

$$e_{N,\infty}^{\mathrm{II}} = \max\{e_{N_1,\infty}^{\mathrm{II}}, e_{N_1,\infty}^{\mathrm{IV}}\} + \sqrt{N_1}g\left(\frac{\pi}{4}\right), \quad m \geq 2,$$

$$e_{N,\infty}^{\mathrm{IV}} = \sqrt{N}\max\left\{g(\varphi)\colon \varphi \in \left(0, \frac{\pi}{4}\right\rangle\right\} + \sqrt{2}e_{N_1,\infty}^{\mathrm{II}} + g\left(\frac{\pi}{4}\right), \quad m \geq 2.$$

•

For the proof of estimates in Theorem 5.5, the factorizations (5.159), (5.160) and Theorem 5.4 are used [59].

Theorem 5.5 yields the following error bounds for several values of $N = 2^m$, $m = 1, 2, \ldots, 6$:

N	$e_{N,2}^{\mathrm{II}}$	$e_{N,\infty}^{\mathrm{II}}$	$e_{N,2}^{\mathrm{IV}}$	$e_{N,\infty}^{\mathrm{IV}}$
2	1.361453	1.060660	1.266694	1.061396
4	3.784973	2.561396	5.070715	4.695545
8	9.050477	6.816865	10.23107	7.702204
16	17.51041	10.70220	19.96457	14.97093
32	32.00139	19.21357	35.07515	22.23433
64	55.18159	28.23423	59.96287	36.77229

The fastly increasing error bounds between exact DCT-II and the corresponding integer DCT-II obtained by algorithm A is caused by the large number of rounding procedures defined by Theorem 5.4.

Consider the integer scaled DCT-II obtained by algorithm B. Let $N = 2^m$, $m \geq 1$, and let $\mathbf{x} \in Z^N$ be an arbitrary input vector. Further, let $\mathbf{y} \in Z^N$ be the resulting integer approximation of $\hat{\mathbf{y}} = \sqrt{N_1}C_N^{\mathrm{II}}\mathbf{x}$ applying algorithm B. Denote again $e_{N,2}^{\mathrm{II}}$ and $e_{N,\infty}^{\mathrm{II}}$ to be the error bounds in the Euclidean and maximum norm, respectively. For the integer approximation $\mathbf{w} \in Z^N$ of $\sqrt{N_1}C_N^{\mathrm{IV}}\mathbf{x}$ via algorithm B, denote the error bounds by $e_{N,2}^{\mathrm{IV}}$ and $e_{N,\infty}^{\mathrm{IV}}$.

Theorem 5.6: (Error bounds estimates between the result of exact scaled DCT-II and the corresponding integer scaled DCT-II obtained by algorithm B in the Euclidean and maximum norm)

Let $N = 2^m$, $m \geq 1$, and let $e_{N,2}^{\mathrm{II}}$, $e_{N,\infty}^{\mathrm{II}}$, $e_{N,2}^{\mathrm{IV}}$ and $e_{N,\infty}^{\mathrm{IV}}$ be the errors occurring, if exact DCT-II output vectors scaled by $\sqrt{N_1}$ are compared with the corresponding integer scaled DCT-II

of algorithm B. Then error bounds in the Euclidean norm can be recursively computed by

$$e_{2,2}^{\mathrm{II}} = \sqrt{h\left(\frac{\pi}{4}\right)}, \quad e_{N,2}^{\mathrm{II}} = \sqrt{(e_{N_{1,2}}^{\mathrm{II}})^2 + (e_{N_{1,2}}^{\mathrm{IV}})^2}, \quad m \geq 2,$$

$$e_{2,2}^{\mathrm{IV}} = \sqrt{h\left(\frac{\pi}{8}\right)}, \quad e_{4,2}^{\mathrm{IV}} = \sqrt{2}\sqrt{h\left(\frac{\pi}{16}\right) + h\left(\frac{3\pi}{16}\right)} + \sqrt{h\left(\frac{\pi}{4}\right)},$$

$$e_{N,2}^{\mathrm{IV}} = \sqrt{N_1}\sqrt{\sum_{k=0}^{\frac{N}{2}-1} h\left(\frac{(2k+1)\pi}{4N}\right)} + 2e_{N_{1,2}}^{\mathrm{II}} + \frac{1}{2}\sqrt{2}, \quad m \geq 3.$$

Error bounds in maximum norm are

$$e_{2,\infty}^{\mathrm{II}} = g\left(\frac{\pi}{4}\right), \quad e_{N,\infty}^{\mathrm{II}} = \max\{e_{N_1,\infty}^{\mathrm{II}}, e_{N_1,\infty}^{\mathrm{IV}}\}, \quad m \geq 2,$$

$$e_{2,\infty}^{\mathrm{IV}} = g\left(\frac{\pi}{8}\right), \quad e_{4,\infty}^{\mathrm{II}} = 2\max\left\{g\left(\frac{\pi}{16}\right), g\left(\frac{3\pi}{16}\right)\right\} + g\left(\frac{\pi}{4}\right),$$

$$e_{N,\infty}^{\mathrm{IV}} = N_1\sqrt{2}\max\left\{g(\varphi)\colon \varphi \in \left(0, \frac{\pi}{4}\right\rangle\right\} + 2e_{N_1,\infty}^{\mathrm{II}} + \frac{1}{2}, \quad m \geq 3.$$

•

Using the factorizations (5.164), (5.165) and Theorem 5.4, the proof of estimates in Theorem 5.6 is similar to the proof of Theorem 5.5 [59].

Theorem 5.6 yields the following error bounds for several values of $N = 2^m$, $m = 1, 2, \ldots, 6$:

N	$e_{N,2}^{\mathrm{II}}$	$e_{N,\infty}^{\mathrm{II}}$	$e_{N,2}^{\mathrm{IV}}$	$e_{N,\infty}^{\mathrm{IV}}$
2	1.361453	1.060660	1.266694	1.061396
4	1.859588	1.061396	3.884236	3.195544
8	4.306432	3.195545	9.466687	8.661157
16	10.40017	8.661157	19.39821	18.96782
32	22.01032	18.96782	41.66265	41.92577
64	47.11932	41.92577	85.03752	86.74255

In particular, the error bounds for 8-point scaled integer DCT-II are reasonably small.

5.5.2.5 A global method to construct invertible integer DCTs

A new global method to derive integer transform from a given arbitrary invertible linear transform has been proposed in Ref. [61]. Principally, the method can be seen as an extension of integer transform with expansion factors. The construction of invertible integer DCT is illustrated for the 8-point DCT-II.

For a general linear transform $\hat{F}$: $R^N \to R^N$ given by $\hat{F}(\mathbf{x}) = H_N\mathbf{x}$, where H_N is an invertible transform matrix, it can be found an invertible integer transform F: $Z^N \to Z^N$ approximating $\hat{F}$ by

$$F(\mathbf{x}) = \text{round}(H_N\mathbf{x}),$$

if H_N satisfies the expansion condition $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N \subseteq H_N((-\frac{1}{2}, \frac{1}{2}\rangle^N)$, where $H_N((-\frac{1}{2}, \frac{1}{2}\rangle^N) = \{H_N\mathbf{r}\colon\ \mathbf{r} \in (-\frac{1}{2}, \frac{1}{2}\rangle^N\}$ is the image of unit cube $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N$ under the linear mapping $\hat{F}$ generated by H_N. However, the transform matrix H_N frequently does not satisfy this condition. In this case we can apply to H_N a suitable expansion factor $\alpha_N > 1$ such that $\alpha_N H_N$ satisfies the expansion condition, i.e., $\alpha_N\, H_N((-\frac{1}{2}, \frac{1}{2}\rangle^N)$ completely covers the unit cube $\langle -\frac{1}{2}, \frac{1}{2}\rangle^N$. Then, an invertible integer transform is simply given by

$$F(\mathbf{x}) = \text{round}(\alpha_N\, H_N\mathbf{x}).$$

This nonlinear integer transform is very close to the exact (scaled) transform $\alpha_N\, H_N\mathbf{x}$ and the error $\alpha_N\, H_N\mathbf{x} - F(\mathbf{x})$ is at most $\frac{1}{2}$ in each component.

The idea is applied to the DCT-II matrix to derive invertible integer DCT-II. Generally, we just need to apply a fast DCT-II algorithm to the input data vector $\mathbf{x}$ and to compute in floating-point arithmetic $\alpha_N\, C_N^{\text{II}}\mathbf{x}$, where α_N is a relatively small constant depending on the value of N. Finally, each DCT-II coefficient is rounded to the nearest integer. The proposed integer transform has two advantages:

1. The fast algorithm being already implemented for the DCT-II can be directly applied.
2. The difference between integer transform and exact (scaled) linear transform is controlled.

Now consider the discrete trigonometric transforms where DCT/DST matrices are obviously orthogonal. But orthogonal matrices do not satisfy the expansion condition and need to be multiplied with a suitable expansion factor. For a given trigonometric transform matrix H_N generating the linear mapping $\hat{F}$: $\mathbf{x} \to H_N\mathbf{x}$ it is necessary to find a minimal expansion factor $\alpha > 0$ such that αH_N satisfies the expansion condition. In the following theorem is shown how to determine the minimal expansion factor for trigonometric transform matrix [61].

Theorem 5.7: (The nonlinear integer mapping for trigonometric transform matrix with the minimal expansion factor)

Let H_N be an invertible matrix and $\hat{F}$: $R^N \to R^N$ with $\hat{F}(\mathbf{x}) = H_N\mathbf{x}$ be the linear mapping generated by H_N. Then the nonlinear mapping $F(\mathbf{x})$: $Z^N \to Z^N$ given by

$$F(\mathbf{x}) = \text{round}(\alpha\, H_N\mathbf{x}),$$

with $\alpha \geq \alpha_N = \|H_N^{-1}\|_\infty$ is an invertible mapping, and we have

$$\mathbf{x} = \text{round}\left(\frac{1}{\alpha} H_N^{-1} F(\mathbf{x})\right).$$

Moreover, it follows that the error is

$$\|\alpha\hat{F}(\mathbf{x}) - F(\mathbf{x})\|_\infty \leq \frac{1}{2},$$

i.e., F is close to the scaled linear mapping $\alpha\hat{F}$.

•

The proof of theorem can be found in Ref. [61]. Analogous invertible mappings can be constructed using other rounding operators such as the *floor* or *ceil* operator.

Specifically, for integer DCT-II the nonlinear invertible mapping $F(\mathbf{x})$: $Z^N \to Z^N$ approximating the DCT-II is defined by

$$F(\mathbf{x}) = \text{round}(\alpha\, C_N^{\text{II}}\, \mathbf{x})$$

with

$$\alpha \geq \alpha_N = \frac{1}{\sqrt{N}} + \frac{1}{\sqrt{2N}}\left(\cot\frac{\pi}{4N} - 1\right).$$

The nonlinear mapping F is invertible, and we have

$$\mathbf{x} = \text{round}\left(\frac{1}{\alpha}[C_N^{\text{II}}]^{\text{T}} F(\mathbf{x})\right) = \text{round}\left(\frac{1}{\alpha}\cdot C_N^{\text{III}} F(\mathbf{x})\right).$$

Moreover, comparing $\hat{\mathbf{x}} = \alpha C_N^{\text{II}}\mathbf{x}$ and $\mathbf{y} = F(\mathbf{x})$ componentwise, we find that

$$|y_j - \hat{x}_j| < \frac{1}{2}, \quad j = 0, 1, \ldots, N-1,$$

i.e., in all components, the nonlinear mapping F rounds the exact (scaled) DCT-II coefficients to the next integer. In particular, the constants α_N satisfy $\alpha_N \leq \sqrt{N}$, i.e., we can replace the normalization factor $\sqrt{\frac{2}{N}}$ in the definition of C_N^{II} by $\sqrt{2}$ and apply a fast DCT-II algorithm to this scaled DCT-II.

Based on the above discussion, the algorithm for integer N-point DCT-II computation is very simple. For integer input vector $\mathbf{x} \in Z^N$ the computation of the forward integer N-point DCT-II consists of the following steps:

1. Compute $\hat{\mathbf{x}} = \alpha\, C_N^{\text{II}}\, \mathbf{x}$ by a fast DCT-II algorithm, where $\alpha \geq \alpha_N$ is chosen suitably, for example, take $\alpha_N = \sqrt{N}$.
2. Compute $\mathbf{y} = \text{round}(\hat{\mathbf{x}})$, where $\mathbf{y} \in Z^N$ approximates $\alpha\, C_N^{\text{II}}\, \mathbf{x}$.

For $N = 8$ we can just take the factor $\alpha = 2\sqrt{2} > \alpha_N$ and use a fast 8-point DCT-II algorithm for $2\sqrt{2}C_N^{\text{II}}\mathbf{x}$.

The inverse integer N-point DCT-II algorithm is equivalently simple and consists of the following steps:

1. Compute $\hat{\mathbf{y}} = \frac{1}{\alpha}[C_N^{\mathrm{II}}]^{\mathrm{T}}\mathbf{y} = \frac{1}{\alpha}\, C_N^{\mathrm{III}}\,\mathbf{y}$ by a fast inverse DCT-II algorithm, where α is chosen as in the forward integer DCT-II algorithm.
2. Compute $\mathbf{x} = \mathrm{round}(\hat{\mathbf{y}})$, where $\mathbf{x} \in Z^N$ is the original input vector.

Using the Theorem 5.7 the integer transforms can be derived for other discrete trigonometric transforms analogously.

5.5.3 Reversible DCTs

Lossless block DCT-II and DCT-IV, called reversible DCTs (RDCTs), have been proposed in Refs. [62–64]. In order to construct the RDCT, the forward and inverse N-point reversible transform is introduced, which is defined by a simple set of equations with associated computational structure. Inserting a nonlinear operation (*round*, *floor* or *ceil*) into the intermediate computational steps, the reversible transform maps integer input signal to integer transform coefficients and integer input signal can be losslessly recovered by the inverse of reversible transform. Essentially, the N-point reversible transform defined by the set of equations is represented by an $N \times N$ generalized transform matrix whose determinant is equal to -1. The elements of generalized transform matrix are derived in the form of algebraic expressions consisting of unknown real-valued coefficients. For a specific N-point DCT its reversible version, N-point RDCT, is obtained by comparing the elements of the $N \times N$ generalized transform matrix with the corresponding elements of a desired $N \times N$ DCT matrix. The process leads to solving the system of linear equations, and the unknown coefficients for the desired DCT matrix can be determined. Having the recursive sparse matrix factorization of DCT matrix the fast N-point RDCT is simply constructed by substituting the lower-order RDCTs, for the corresponding lower-order block DCTs which make up the N-point DCT. If the nonlinear *floor* operation is ignored, then the RDCT becomes the original real-valued DCT with determinant ± 1. The normalized N-point RDCT can also be used to avoid the problem that the dynamic range of transform coefficients is nonuniform.

Since the N-point RDCT is represented by the $N \times N$ generalized transform matrix, the method can be applied to any other discrete trigonometric transform using the (recursive) factorization of a given DCT/DST matrix.

5.5.3.1 Matrix factorizations of 2-point block transforms

Let A be a matrix of order 2 with elements $a, b, c, d \in R$, $b \neq 0$ and with $\det(A) = +1$. Then, the matrix A and its inverse A^{-1} can be respectively factorized into the following matrix products [7]:

$$A = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 1 & 0 \\ \frac{d-1}{b} & 1 \end{pmatrix} \begin{pmatrix} 1 & b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \frac{a-1}{b} & 1 \end{pmatrix},$$

$$A^{-1} = \begin{pmatrix} 1 & 0 \\ -\frac{d-1}{b} & 1 \end{pmatrix} \begin{pmatrix} 1 & -b \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\frac{a-1}{b} & 1 \end{pmatrix}. \tag{5.167}$$

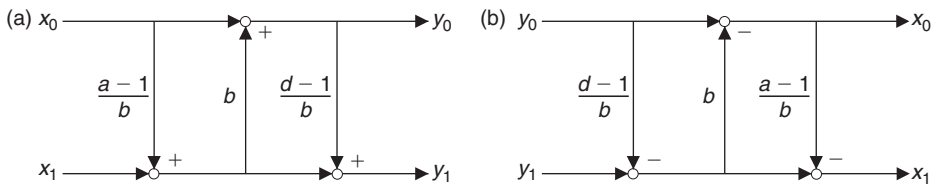


Fig. 5.36. The computational structures for the 2-point block transform represented by the matrix A: (a) forward and (b) inverse.

The matrices A and A^{-1} represent forward and inverse 2-point block transforms, respectively. The corresponding computational structures for the forward and inverse 2-point block transform are shown in Fig. 5.36(a) and (b), respectively. Actually, the factorization of matrix A given by (5.167) can be obtained by the *PLUS* factorization algorithm described in Section 5.2.7. Substituting $a = d = \cos\varphi$, $b = -\sin\varphi$ and $c = \sin\varphi$ into factorization (5.167) we get the well-known LUL factorization of Givens–Jacobi rotation G_φ and $G_{-\varphi}$.

Let B be a matrix of order 2 with elements $a, b, c, d \in R$, $a \neq 0$ and with $\det(B) = -1$. Then, the matrix B can be factorized into the following matrix product [63, 64]:

$$B = \begin{pmatrix} a & b \\ c & d \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & \frac{c-1}{a} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & a \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & \frac{b-1}{a} \end{pmatrix}. \tag{5.168}$$

Since the inverse of each factored matrix in (5.168) is given by

$$\begin{pmatrix} 0 & 1 \\ 1 & s \end{pmatrix}^{-1} = \begin{pmatrix} -s & 1 \\ 1 & 0 \end{pmatrix},$$

then the factorization of matrix B^{-1} has the following form:

$$B^{-1} = \begin{pmatrix} -\frac{b-1}{a} & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -a & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} -\frac{c-1}{a} & 1 \\ 1 & 0 \end{pmatrix}. \tag{5.169}$$

The factorization of matrix B given by (5.168) can be also obtained by *PLUS* factorization algorithm described in Section 5.2.7. Matrix factors are of the same type with opposite unit main diagonal. The corresponding computational structures for the forward and inverse 2-point block transform are shown in Fig. 5.37(a) and (b), respectively.

Note: Since $J_2 J_2 = I_2$ and $\det(J_2) = -1$, the factorization of matrix B given by (5.168) can be flexibly modified by row and column permutations in factored matrices to get different but equivalent factorizations defining the same computational structure which is shown in Fig. 5.37.

The factorization of matrix B given by (5.168) is essential to derive N-point reversible transform.

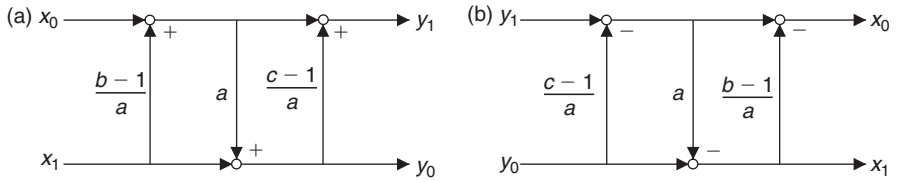


Fig. 5.37. Computational structures for the 2-point block transform represented by the matrix B: (a) forward and (b) inverse.

5.5.3.2 N-point reversible transform

2-point block transform represented by the matrix B with the factorization given by (5.168) applied to integer input vector $\mathbf{x} = [x_0, x_1]^{\mathrm{T}}$ can be expressed as

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = \begin{pmatrix} 0 & 1 \\ 1 & \frac{c-1}{a} \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & a \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & \frac{b-1}{a} \end{pmatrix} \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \tag{5.170}$$

where y_1 and y_2 are real-valued coefficients. Equation (5.170) is equivalent to the following set of equations [63, 64]:

$$\begin{aligned} y_0 &= x_0 + k_0 x_1, \\ y_1 &= x_1 + k_1 y_0, \\ y_2 &= y_0 + k_2 y_1, \quad x_0, x_1 \in Z, \quad y_0, y_1, y_2 \in R \text{ and } k_0, k_1, k_2 \in R, \end{aligned} \tag{5.171}$$

where $k_0 = \frac{b-1}{a}$, $k_1 = a$ and $k_2 = \frac{c-1}{a}$, and y_1 and y_2 are transform coefficients. The 2-point block transform defined by (5.171) can be represented by the generalized transform matrix T_2 as

$$\begin{pmatrix} y_1 \\ y_2 \end{pmatrix} = T_2 \begin{pmatrix} x_0 \\ x_1 \end{pmatrix}, \quad T_2 = \begin{pmatrix} k_1 & 1 + k_0 k_1 \\ 1 + k_1 k_2 & k_0 + k_2(1 + k_0 k_1) \end{pmatrix}. \tag{5.172}$$

It is easy to verify that the generalized transform matrix T_2 has a determinant equal to -1. For a specific 2-point block transform, the unknown coefficients k_0, k_1 and k_2 are obtained by comparing the elements of the generalized transform matrix T_2 with the corresponding elements of a desired 2×2 transform matrix (e.g., C_2^{II} or C_2^{IV}). Introducing the *floor* function the equation (5.171) becomes

$$\begin{aligned} y_0 &= x_0 + \left\lfloor k_0 x_1 + \frac{1}{2} \right\rfloor, \\ y_1 &= x_1 + \left\lfloor k_1 y_0 + \frac{1}{2} \right\rfloor, \\ y_2 &= y_0 + \left\lfloor k_2 y_1 + \frac{1}{2} \right\rfloor, \quad x_0, x_1, y_0, y_1, y_2 \in Z \text{ and } k_0, k_1, k_2 \in R. \end{aligned} \tag{5.173}$$

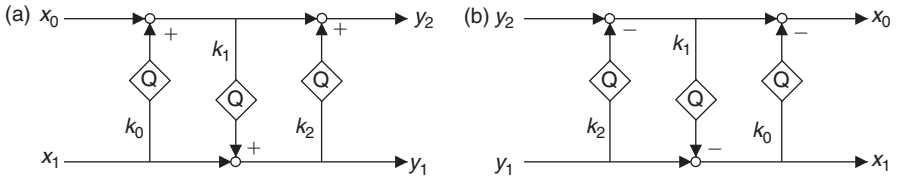


Fig. 5.38. Computational structures for the 2-point reversible transform: (a) forward and (b) inverse. Q denotes the *floor* function.

Equation (5.173) is reversible, i.e., x_0 and x_1 are losslessly reconstructed from y_1 and y_2 as

$$\begin{aligned} y_0 &= y_2 - \left\lfloor k_2 y_1 + \frac{1}{2} \right\rfloor, \\ x_1 &= y_1 - \left\lfloor k_1 y_0 + \frac{1}{2} \right\rfloor, \\ x_0 &= y_0 - \left\lfloor k_0 x_1 + \frac{1}{2} \right\rfloor. \end{aligned} \tag{5.174}$$

The computational structures for the forward and inverse 2-point reversible transform are shown in Fig. 5.38(a) and (b), respectively. Q denotes the *floor* function.

The 2-point reversible transform can be generalized to an N-point reversible transform as [62–64]

$$\begin{aligned} y_j &= x_j + \left\lfloor \sum_{i=0}^{j-1} k_{ij} y_i + \sum_{i=j+1}^{N-1} k_{ij} x_i + \frac{1}{2} \right\rfloor, \quad j = 0, 1, \ldots, N-1, \\ y_N &= y_0 + \left\lfloor \sum_{i=1}^{N-1} k_{iN} y_i + \frac{1}{2} \right\rfloor, \quad k_{ij} \in R, \end{aligned} \tag{5.175}$$

where $\sum_{i=0}^{-1} = \sum_{i=N}^{N-1} \equiv 0$, and $y_1, y_2, \ldots, y_N \in Z$ are transform coefficients and $x_i, i = 0, \ldots, N-1$ are input vector components. The inverse N-point reversible transform is then defined as

$$\begin{aligned} y_0 &= y_N - \left\lfloor \sum_{i=1}^{N-1} k_{iN} y_i + \frac{1}{2} \right\rfloor, \\ x_j &= y_j - \left\lfloor \sum_{i=0}^{j-1} k_{ij} y_i + \sum_{i=j+1}^{N-1} k_{ij} x_i + \frac{1}{2} \right\rfloor, \quad j = 0, 1, \ldots, N-1, \; k_{ij} \in R. \end{aligned} \tag{5.176}$$

In Section 5.2.9 the determinants of DCT/DST matrices have been evaluated. For $N = 2$ all the DCT and DST matrices are non-eigenorthogonal, i.e., their determinant is equal to -1

while for $N > 2$ almost all DCT and DST matrices are eigenorthogonal with determinant equal to $+1$. The N-point reversible transform can be represented by $N \times N$ generalized transform matrix T_N with determinant -1. In the case that determinants of T_N and of DCT/DST matrix differ, the construction of N-point RDCT/RDST (reversible DST) transform can be realized without any problem. Comparing the corresponding elements of the generalized transform matrix T_N and the desired $N \times N$ DCT/DST matrix we determine the values of unknown coefficients k_{ij}. If the determinant of the desired DCT/DST matrix is $+1$ then substituting the obtained coefficients k_{ij} for elements of T_N its determinant will also have the value of $+1$, and elements in the last row of T_N will have opposite signs in what is equivalent to multiplying the last row by -1 [63]. It indicates that the generalized transform matrix T_N is adaptable according to the value of the determinant of the desired transform matrix. Consequently, the last transform coefficient will have the value with opposite sign. In fact, this can be observed in the construction of the 4-point RDCT-IV.

Consider the 4-point reversible transform defined by (5.175). For $N = 4$ we obtain the set of equations defining the forward 4-point reversible transform as

$$\begin{aligned}
y_0 &= x_0 + \left\lfloor k_{10}x_1 + k_{20}x_2 + k_{30}x_3 + \frac{1}{2} \right\rfloor, \\
y_1 &= x_1 + \left\lfloor k_{01}y_0 + k_{21}x_2 + k_{31}x_3 + \frac{1}{2} \right\rfloor, \\
y_2 &= x_2 + \left\lfloor k_{02}y_0 + k_{12}y_1 + k_{32}x_3 + \frac{1}{2} \right\rfloor, \\
y_3 &= x_3 + \left\lfloor k_{03}y_0 + k_{13}y_1 + k_{23}y_2 + \frac{1}{2} \right\rfloor, \\
y_4 &= y_0 + \left\lfloor k_{14}y_1 + k_{24}y_2 + k_{34}y_3 + \frac{1}{2} \right\rfloor,
\end{aligned} \tag{5.177}$$

where $y_1, y_2, y_3, y_4 \in Z$ are transform coefficients. The 4-point reversible transform can be represented by the generalized transform matrix T_4 given by

$$\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = T_4 \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}, \quad T_4 = \begin{pmatrix} t_{11} & t_{12} & t_{13} & t_{14} \\ t_{21} & t_{22} & t_{23} & t_{24} \\ t_{31} & t_{32} & t_{33} & t_{34} \\ t_{41} & t_{42} & t_{43} & t_{44} \end{pmatrix}, \tag{5.178}$$

where

$$\begin{aligned}
t_{11} &= k_{01}, \\
t_{12} &= 1 + k_{01}k_{10}, \\
t_{13} &= k_{21} + k_{01}k_{20}, \\
t_{14} &= k_{31} + k_{01}k_{30},
\end{aligned}$$

$$
\begin{aligned}
t_{21} &= k_{02} + k_{12}t_{11},\\
t_{22} &= k_{02}k_{10} + k_{12}t_{12},\\
t_{23} &= 1 + k_{02}k_{20} + k_{12}t_{13},\\
t_{24} &= k_{02}k_{30} + k_{12}t_{14} + k_{32},
\end{aligned}
$$

$$
\begin{aligned}
t_{31} &= k_{03} + k_{13}t_{11} + k_{23}t_{21},\\
t_{32} &= k_{03}k_{10} + k_{13}t_{12} + k_{32}t_{22},\\
t_{33} &= k_{03}k_{20} + k_{13}t_{13} + k_{32}t_{23},\\
t_{34} &= 1 + k_{03}k_{30} + k_{13}t_{14} + k_{32}t_{24},
\end{aligned}
$$

$$
\begin{aligned}
t_{41} &= 1 + k_{14}t_{11} + k_{24}t_{21} + k_{34}t_{31},\\
t_{42} &= k_{10} + k_{14}t_{12} + k_{24}t_{22} + k_{34}t_{32},\\
t_{43} &= k_{20} + k_{14}t_{13} + k_{24}t_{23} + k_{34}t_{33},\\
t_{44} &= k_{30} + k_{14}t_{14} + k_{24}t_{24} + k_{34}t_{34}.
\end{aligned}
$$

It can be verified that the determinant of generalized transform matrix T_4 is equal to -1. For a specific 4-point block transform the unknown coefficients k_{ij} are obtained by comparing the elements of the generalized transform matrix T_4 with the corresponding elements of a desired transform matrix of order 4 (e.g., C_4^{IV}). The computational structures for the forward and inverse 4-point reversible transform are shown in Fig. 5.39(a) and (b), respectively. Q denotes the *floor* function. The computational structure in Fig. 5.39(a) defines the following factorization of the generalized transform matrix T_4:

$$
\begin{aligned}
T_4 = &\begin{pmatrix} 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\\ 1 & k_{14} & k_{24} & k_{34} \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ k_{03} & k_{13} & k_{23} & 1 \end{pmatrix}
\begin{pmatrix} 1 & 0 & 0 & 0\\ 0 & 1 & 0 & 0\\ k_{02} & k_{12} & 1 & k_{32}\\ 0 & 0 & 0 & 1 \end{pmatrix}\\
&\times \begin{pmatrix} 1 & 0 & 0 & 0\\ k_{01} & 1 & k_{21} & k_{31}\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 \end{pmatrix}
\begin{pmatrix} 1 & k_{10} & k_{20} & k_{30}\\ 0 & 1 & 0 & 0\\ 0 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 \end{pmatrix}.
\end{aligned}
\tag{5.179}
$$

It is interesting to note that the factored matrices in (5.179) are the so-called single-row elementary matrices introduced in Ref. [9].

In the following section the construction of fast RDCT is illustrated for 8-point DCT–II.

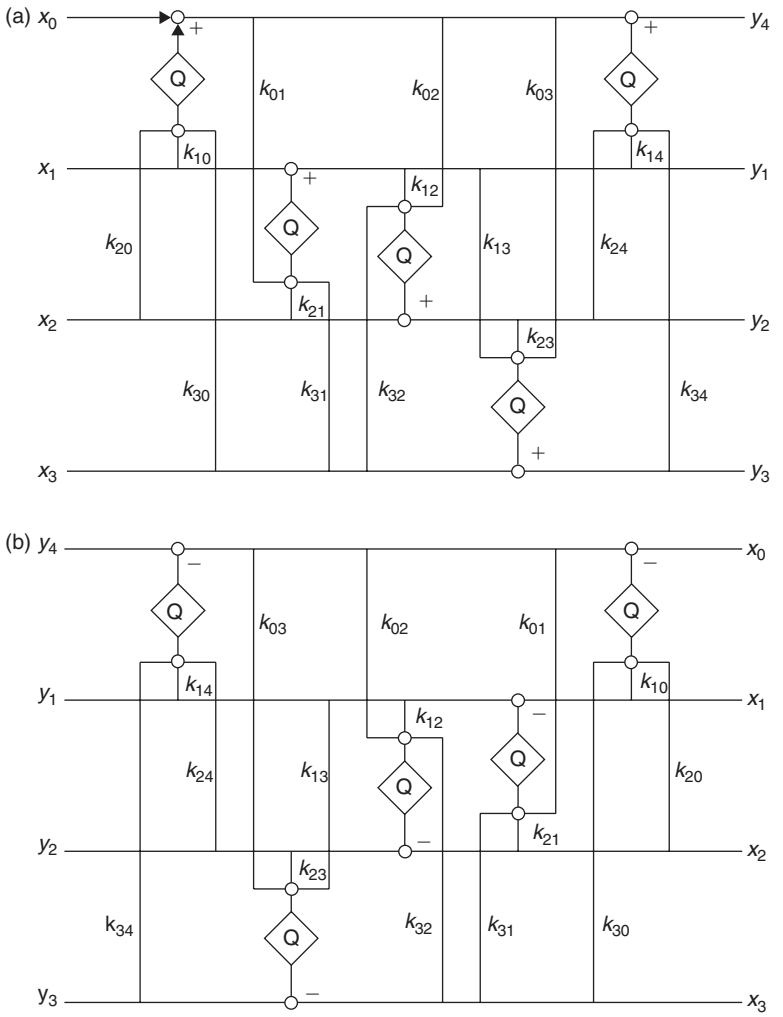


Fig. 5.39. Computational structures for the 4-point reversible transform: (a) forward and (b) inverse. Q denotes the *floor* function.

5.5.3.3 The fast 8-point RDCT-II

From the modified EOT factorization of matrix $\hat{C}_8^{\mathrm{II}}$ given by (5.135) and (5.136) applied recursively we have

$$\hat{C}_8^{\mathrm{II}} = \frac{\sqrt{2}}{2} \begin{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & C_2^{\mathrm{IV}} J_2 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & 0 \\ 0 & C_4^{\mathrm{IV}} J_4 \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.180}$$

where the non-eigenorthogonal matrices C_2^{II}, C_2^{IV} and the eigenorthogonal C_4^{IV} matrix are given by

$$C_2^{\mathrm{II}} = \begin{pmatrix} \frac{\sqrt{2}}{2} & \frac{\sqrt{2}}{2} \\ \frac{\sqrt{2}}{2} & -\frac{\sqrt{2}}{2} \end{pmatrix}, \quad \det(C_2^{\mathrm{II}}) = -1,$$

$$C_2^{\mathrm{IV}} J_2 = \begin{pmatrix} \cos\frac{\pi}{8} & \sin\frac{\pi}{8} \\ \sin\frac{\pi}{8} & -\cos\frac{\pi}{8} \end{pmatrix} J_2, \quad \det(C_2^{\mathrm{IV}}) = \det(J_2) = -1, \ \ \det(C_2^{\mathrm{IV}} J_2) = +1,$$

$$C_4^{\mathrm{IV}} J_4 = \frac{\sqrt{2}}{2} \begin{pmatrix} \cos\frac{\pi}{16} & \cos\frac{3\pi}{16} & \sin\frac{3\pi}{16} & \sin\frac{\pi}{16} \\ \cos\frac{3\pi}{16} & -\sin\frac{\pi}{16} & -\cos\frac{\pi}{16} & -\sin\frac{3\pi}{16} \\ \sin\frac{3\pi}{16} & -\cos\frac{\pi}{16} & \sin\frac{\pi}{16} & \cos\frac{3\pi}{16} \\ \sin\frac{\pi}{16} & -\sin\frac{3\pi}{16} & \cos\frac{3\pi}{16} & -\cos\frac{\pi}{16} \end{pmatrix} J_4,$$

$$\det(C_4^{\mathrm{IV}}) = +1, \quad \det(J_4) = +1 \quad \text{and} \quad \det(C_4^{\mathrm{IV}} J_4) = +1. \tag{5.181}$$

According to the modified EOT factorization, the matrix $\hat{C}_8^{\mathrm{II}}$ is decomposed into 2- and 4-point DCT-II and DCT-IV block transforms. In order to obtain the 8-point RDCT-II we need to compare the elements of the generalized matrix T_2 in (5.172) with the corresponding elements of the matrices C_2^{II} and C_2^{IV}, and then to compare the elements of generalized matrix T_4 in (5.178) with the corresponding elements of the matrix C_4^{IV}.

For the 2-point RDCT-II we get the values of coefficients k_0, k_1 and k_2 as

$$k_1 = \frac{\sqrt{2}}{2}, \quad k_0 = k_2 = 1 - \sqrt{2},$$

and for the 2-point RDCT-IV we get the following values of coefficients k_0, k_1 and k_2 as

$$k_1 = \cos\frac{\pi}{8}, \quad k_0 = k_2 = \frac{\sin\frac{\pi}{8} - 1}{\cos\frac{\pi}{8}}.$$

Finally, comparing the corresponding elements of generalized transform matrix T_4 in (5.178) and the matrix C_4^{IV} we get the values of the coefficients k_{ij} for the 4-point RDCT-IV as

$$k_{01} = \frac{\sqrt{2}}{2}\cos\frac{\pi}{16},$$

$$k_{10} = \frac{\cos\frac{3\pi}{16} - \sqrt{2}}{\cos\frac{\pi}{16}},$$

$$k_{12} = -\frac{\sqrt{2}}{2}\left[\sin\frac{\pi}{16} + k_{10}\cos\frac{3\pi}{16}\right],$$

$$k_{02} = \frac{\sqrt{2}}{2}\left[\cos\frac{3\pi}{16} - k_{12}\cos\frac{\pi}{16}\right],$$

$$k_{20} = -\frac{\frac{\sqrt{2}}{2}\left[\cos\frac{\pi}{16} + k_{12}\sin\frac{3\pi}{16}\right]}{k_{02}},$$

$$k_{21} = \frac{\sqrt{2}}{2}\left[\sin\frac{3\pi}{16} - k_{20}\cos\frac{\pi}{16}\right],$$

$$k_{32} = k_{21},$$

$$k_{30} = -\frac{\frac{\sqrt{2}}{2}\left[\sin\frac{3\pi}{16} + k_{12}\sin\frac{\pi}{16}\right] - k_{12}}{k_{02}},$$

$$k_{31} = \frac{\sqrt{2}}{2}\left[\sin\frac{\pi}{16} - k_{30}\cos\frac{\pi}{16}\right],$$

$$k_{03} = -k_{01},$$

$$k_{13} = -k_{31},$$

$$k_{23} = k_{21},$$

$$k_{14} = -k_{30},$$

$$k_{24} = k_{20},$$

$$k_{34} = -k_{10}.$$

Substituting the 2- and 4-point RDCTs-II and RDCTs-IV for the corresponding 2- and 4-point DCT-II and DCT-IV block transforms which make up the 8-point DCT-II transform in (5.180) we obtain the fast 8-point RDCT-II which is shown in Fig. 5.40. The fast 4-point RDCT-II is shown in a dotted box. Since $\det(C^{\mathrm{IV}}) = +1$ and $\det(T_4) = -1$, then for given coefficients k_{ij} in the implementation of 4-point RDCT-IV by computational structure shown in Fig. 5.39 the last transform coefficient y_4 has the opposite sign, i.e., $y_4 = -y_4$. The normalization of output RDCT-II coefficients is given by the diagonal matrix $\hat{D}_8 = \mathrm{diag}\{\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}\}$.

The concept of normalized DCT-II is based on the orthogonal EOT matrix factorization of DCT-II matrix. The orthogonal EOT factorization of $\hat{C}_8^{\mathrm{II}}$ is defined as

$$\hat{C}_8^{\mathrm{II}} = \begin{pmatrix} \begin{pmatrix} C_2^{\mathrm{II}} & 0 \\ 0 & C_2^{\mathrm{IV}} J_2 \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix} & \begin{matrix} 0 \\ 0 \end{matrix} \\ \begin{matrix} 0 & 0 \end{matrix} & C_4^{\mathrm{IV}} J_4 \end{pmatrix} \frac{\sqrt{2}}{2} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.182}$$

where matrices C_2^{II}, C_2^{IV} and C_4^{IV} are given by (5.181). Compared to (5.180) in the orthogonal EOT factorization of $\hat{C}_8^{\mathrm{II}}$, all factored matrices including butterfly matrices

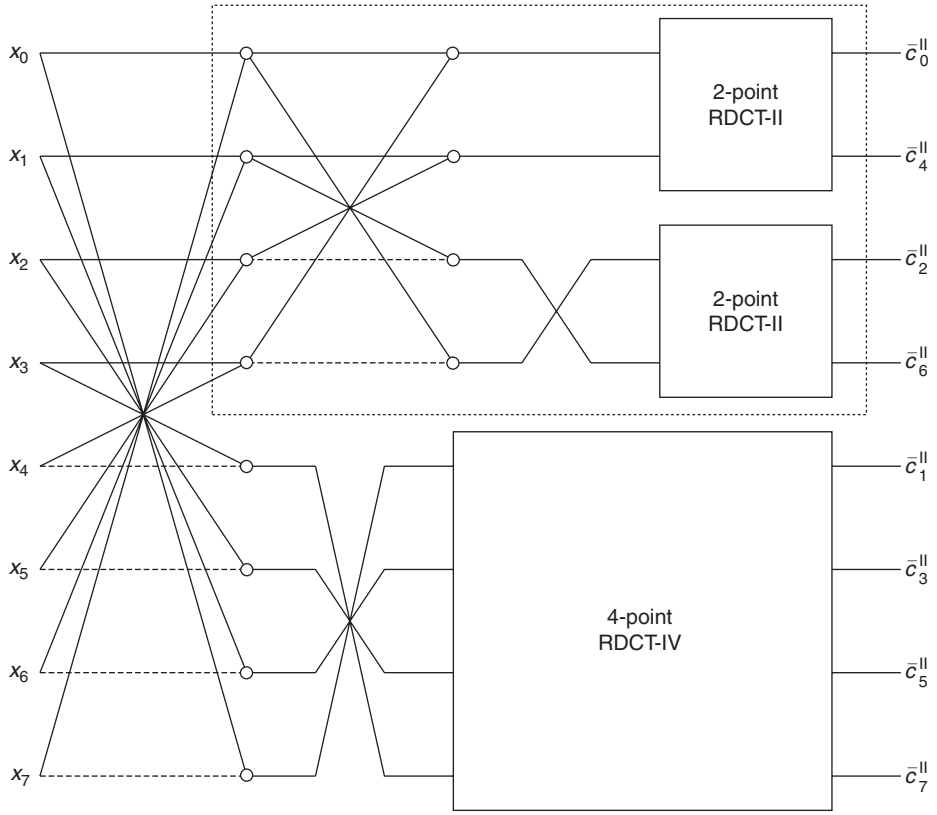


Fig. 5.40. The fast 4- and 8-point RDCTs-II.

are orthogonal and consequently no normalization is needed. Thus, the dynamic range of the output transform coefficients can be kept uniform. By suitable permutations, the orthogonal EOT factorization of the $\hat{C}_8^{II}$ matrix can be decomposed into 2-point DCT-II block transforms and 2- and 4-point DCT-IV block transforms only. Replacing each 2- and 4-point DCT-II and DCT-IV block transforms by the corresponding 2- and 4-point RDCTs-II and RDCTs-IV, respectively, the 8-point normalized RDCT-II is obtained. The fast 8-point normalized RDCT-II is shown in Fig. 5.41. The fast 4-point normalized RDCT-II is shown in a dotted box.

The inverse 8-point RDCT-II, RDCT-III, is obtained by reversing the forward 8-point RDCT-II and performing the inverse 2- and 4-point RDCTs-II and RDCTs-IV, respectively.

5.5.4 Signed DCT square wave transform

An efficient square wave transform called the signed DCT-II (SignDCT-II) has been proposed in Ref. [65]. The SignDCT-II is obtained by applying the signum function operator to the elements of the DCT-II matrix C_N^{II}, i.e., the SignDCT-II matrix of order N

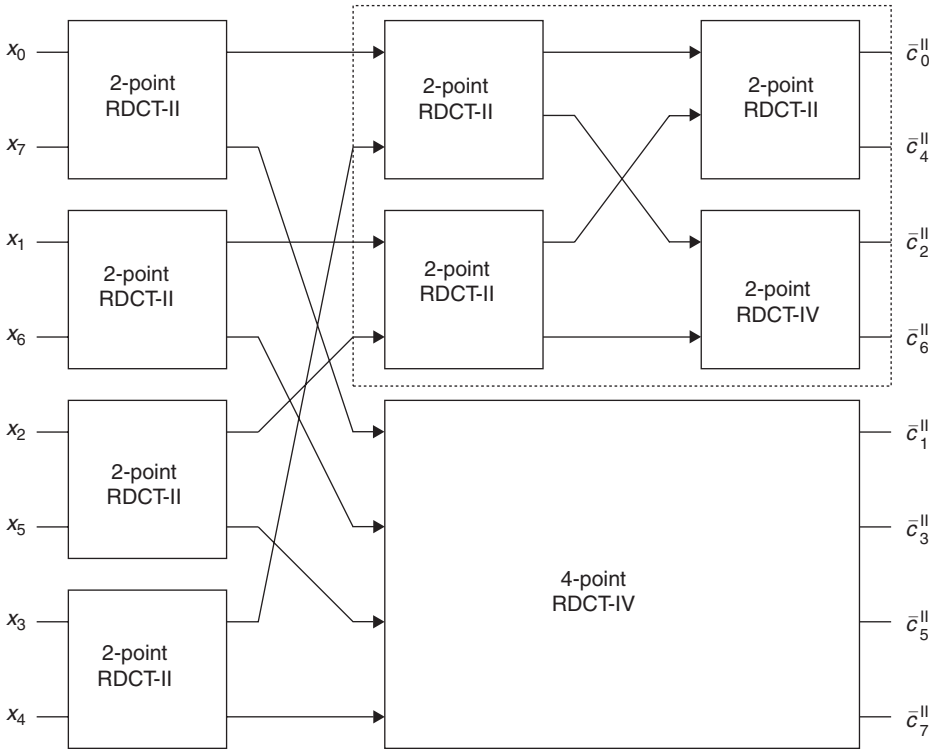


Fig. 5.41. The fast 4- and 8-point normalized RDCTs-II.

denoted by $C_N^{\text{SignDCT-II}}$ is defined as

$$\left[C_N^{\text{SignDCT-II}}\right]_{ij} = \text{sign}\left(\cos\frac{\pi(2j+1)i}{2N}\right), \quad i,j = 0,1,\ldots,N-1, \tag{5.183}$$

where

$$\text{sign}(x) = \begin{cases} +1 & x > 0, \\ 0 & x = 0, \\ -1 & x < 0. \end{cases}$$

Several advantages are immediately apparent for the SignDCT-II. All the elements are ± 1 and no multiplication is required, i.e., it is multiplierless. Transform order of the SignDCT-II need not to be a specific integer such as a power of 2. The SignDCT-II maintains the periodicity and spectral properties of the DCT-II, and consequently, it has good decorrelation and energy compaction characteristics. This fact can be verified for a given value of N by comparing spectra of the DCT-II and SignDCT-II that are very close to each other [65].

Using the definition of SignDCT-II for $N = 8$ its transform matrix $C_8^{\text{SignDCT-II}}$ is given by

$$C_8^{\text{SignDCT-II}} = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & 1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & -1 & 1 & 1 & -1 & -1 & 1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{pmatrix}, \tag{5.184}$$

and the $\det(C_8^{\text{SignDCT-II}}) = \frac{1}{2}$. Since the SignDCT-II is derived from the DCT-II, any fast algorithm for the DCT-II computation can be adopted for the efficient SignDCT-II computation. This is valid for the forward SignDCT-II only, because the matrix $C_8^{\text{SignDCT-II}}$ is not orthogonal. However, it is invertible and its inverse is given by

$$\left[C_8^{\text{SignDCT-II}}\right]^{-1} = \frac{1}{\sqrt{8}} \begin{pmatrix} 1 & 2 & 1 & 2 & 1 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & -1 & -2 & -1 & 0 \\ 1 & 0 & -1 & -2 & -1 & 0 & 1 & 2 \\ 1 & 0 & -1 & 0 & 1 & 2 & -1 & -2 \\ 1 & 0 & -1 & 0 & 1 & -2 & -1 & 2 \\ 1 & 0 & -1 & 2 & -1 & 0 & 1 & -2 \\ 1 & -2 & 1 & 0 & -1 & 2 & -1 & 0 \\ 1 & -2 & 1 & -2 & 1 & 0 & 1 & 0 \end{pmatrix}. \tag{5.185}$$

Unfortunately, these specialized features of the SignDCT-II are not valid for all orders. In order to obtain the fast multiplierless 8-point SignDCT-II, we utilize the EOT factorization defined by (4.17). Let $\hat{C}_8^{\text{SignDCT-II}}$ be the SignDCT-II matrix with its rows rearranged so that the upper half consists of even-indexed rows followed by the lower half of odd-indexed rows, both in bit-reversed order. Then, according to EOT factorization the matrix $\hat{C}_8^{\text{SignDCT-II}}$ can be recursively factorized as

$$\hat{C}_8^{\text{SignDCT-II}} = \frac{1}{\sqrt{8}} \begin{pmatrix} \hat{C}_4^{\text{SignDCT-II}} & 0 \\ 0 & \begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{matrix} \end{pmatrix} \begin{pmatrix} I_4 & J_4 \\ J_4 & -I_4 \end{pmatrix}, \tag{5.186}$$

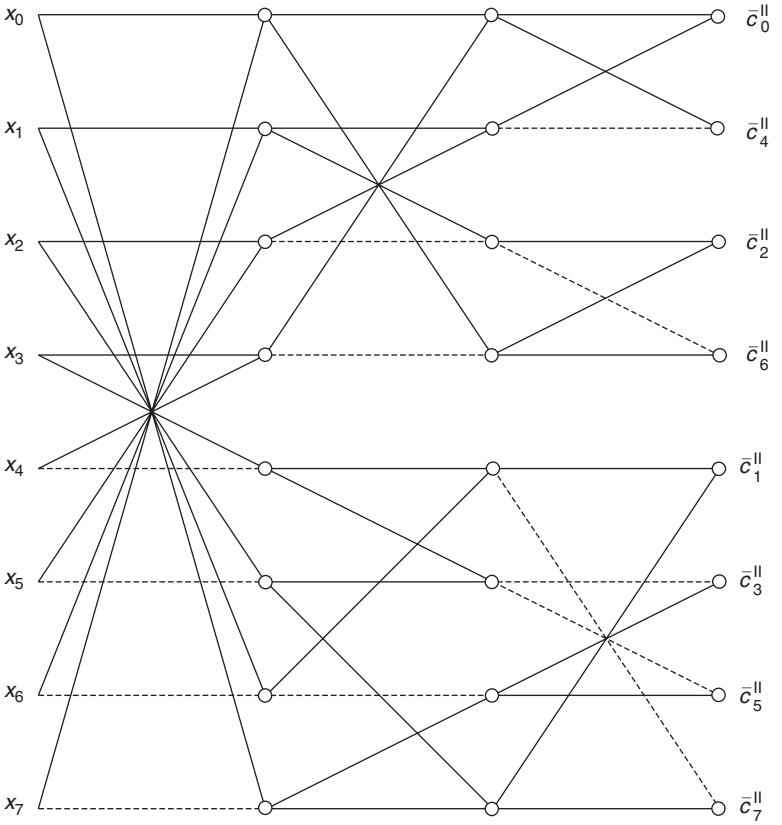


Fig. 5.42. The fast forward multiplierless 8-point SignDCT-II.

where

$$\hat{C}_4^{\text{SignDCT-II}} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{pmatrix} \begin{pmatrix} I_2 & J_2 \\ J_2 & -I_2 \end{pmatrix},$$

$$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & 1 \\ -1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & -1 & 1 & 0 \\ -1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 0 & -1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix}. \tag{5.187}$$

The corresponding fast forward multiplierless 8-point SignDCT-II is shown in Fig. 5.42. Its computational complexity is 24 additions.

5.6 Late additions with comments

The integer approximation of discrete sinusoidal transforms has recently been an active research topic. The recently established international coding standards such as JPEG2000, MPEG recommend the use of integer transforms for lossless signal coding applications. For example, MPEG is considering the need for development of a new voluntary standard specifying a fixed-point approximation to the ideal inverse and forward 8×8 DCT-II (see document ISO/IEC/SC29/WG11/N6915, Hong Kong, January 2005). This document provides all details including evaluation criteria, developing the inverse and forward 8×8 IntDCT that can meet the evaluation criteria and integrating with MPEG codecs.

One can observe that the results of linear algebra have played an important role in the integer approximation of transforms. In fact, matrix factorizations are powerful mathematical tools to construct invertible/reversible transforms with integer-to-integer mapping from the corresponding linear transform matrices. At the time of finalizing this book several articles have appeared in the refereed journals and conferences which are related to improving the integer approximation of transforms [66–74]. Therefore, in this section we briefly comment on the recent important developments. Reader can find the details in references.

5.6.1 PL_1UL_2 factorization of transform matrices

In Ref. [66] it has been shown that a linear transform defined by an invertible matrix T_N of order N with a determinant of unit magnitude can be factorized into a product of unit triangular matrices with diagonal elements ± 1 as

$$T_N = P\,L_1\,U\,L_2, \tag{5.188}$$

where P is a permutation matrix, L_1 and L_2 are unit lower triangular matrices and U is a unit upper triangular matrix. Matrices L_1, U and L_2 are reversibly implemented by matrix–vector multiplications in floating-point arithmetic with results immediately rounded to the nearest integer. The factorization (5.188) is actually a *PLUS* factorization ($L_2 = S$) of T_N described in Section 5.5.1 for the DCT-II matrix C_8^{II}.

5.6.2 The normalized integer transforms

When an integer transform is used for lossless coding applications, it has to be normalized to preserve the energy of the input signal in the frequency domain [70]. The normalized integer transform is associated with the orthogonal (recursive) factorization of the corresponding transform matrix. As an example, see the recursive EOT orthogonal factorization of C_8^{II} matrix given by (5.182) in Section 5.5.3 (reversible DCTs). All trivial butterflies are orthogonal. Rotation matrices $H_{\frac{\pi}{4}}$ or equivalently $G_{-\frac{\pi}{4}}\,J_2$ are computed by using the LUL (ULU) structure, where each scalar multiplication is followed by a rounding operation. Consequently, more multiplications are needed and thus the computational complexity both of noninteger and approximated integer transform increases. As well, due

to the realization of trivial orthogonal butterflies the error caused by integer approximation increases too. In case of long transform sizes this approximation error in the frequency domain becomes considerable. The total accumulated approximation error has a significant impact on the lossless coding efficiency. In order to improve the integer transform it is desirable to minimize not only the computational complexity and the number of rounding operations but also the approximation error as much as possible [68–70]. With the aim to improve integer transforms in terms of the computational efficiency and reducing the approximation error, two schemes have been presented in Ref. [70]:

1. The first scheme is based on the employment of two-step. DLU structure instead of three-step LUL (ULU) structure. Consequently, the number of multipliers to be approximated in trivial orthogonal butterflies is reduced.
2. The second scheme is based on the so-called multidimensional (MDL) computational structure defined by the factorization of a diagonal block matrix into the product of three block matrices [68–70]. The MDL computational structure significantly reduces both the number of rounding operations and the approximation error.

5.6.3 The MDL computational structure

The block matrices of the form:

$$\begin{pmatrix} \pm I_N & 0 \\ T_N & \pm I_N \end{pmatrix}, \quad \begin{pmatrix} \pm I_N & T_N \\ 0 & \pm I_N \end{pmatrix}, \tag{5.189}$$

where I_N is the identity matrix and T_N is the square nonsingular matrix both of order N, are respectively called the lower and upper quasi-triangular matrices [3]. In particular, the block matrix of the form:

$$\begin{pmatrix} A_N & 0 \\ 0 & B_N \end{pmatrix}, \tag{5.190}$$

where A_N and B_N are square nonsingular matrices, is called a quasi-diagonal matrix [3]. Generally, the algebra of square (quasi-triangular) block matrices such as addition and multiplication is similar to that of square (triangular) matrices with scalar elements. In particular [3]:

- The product of two lower (upper) quasi-triangular matrices is lower (upper) quasi-triangular matrix.
- The determinant of quasi-triangular matrix (and quasi-diagonal matrix too) with square diagonal blocks is equal to the product of determinants of diagonal square block matrices.
- The transposition of lower (upper) quasi-triangular matrix is upper (lower) quasi-triangular matrix.

Obviously, the inverses of quasi-triangular matrices given by (5.189) exist and they can be simply obtained as

$$\begin{pmatrix} I_N & 0 \\ T_N & I_N \end{pmatrix}^{-1} = \begin{pmatrix} I_N & 0 \\ -T_N & I_N \end{pmatrix}, \quad \begin{pmatrix} -I_N & 0 \\ T_N & I_N \end{pmatrix}^{-1} = \begin{pmatrix} -I_N & 0 \\ T_N & I_N \end{pmatrix},$$
$$\begin{pmatrix} I_N & 0 \\ T_N & -I_N \end{pmatrix}^{-1} = \begin{pmatrix} I_N & 0 \\ T_N & -I_N \end{pmatrix}, \quad \begin{pmatrix} -I_N & 0 \\ T_N & -I_N \end{pmatrix}^{-1} = \begin{pmatrix} -I_N & 0 \\ -T_N & -I_N \end{pmatrix}. \tag{5.191}$$

The MDL computational structure is based on the factorization of 2×2 diagonal scaling matrix with determinant equal to $+1$ into the product of three unit lower and unit upper triangular matrices as follows [67]:

$$\begin{aligned} \begin{pmatrix} d & 0 \\ 0 & d^{-1} \end{pmatrix} &= \begin{pmatrix} -1 & 0 \\ d^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \\ 1 & d^{-1} \end{pmatrix} \\ &= \begin{pmatrix} -1 & 0 \\ d^{-1} & 1 \end{pmatrix} \begin{pmatrix} 1 & -d \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ d^{-1} & 1 \end{pmatrix} J_2, \quad d \in R, \ d \neq 0. \end{aligned} \tag{5.192}$$

The factorization (5.192) provides the basic idea to derive the MDL computational structure. It can be extended to cases where each scalar element in the factored matrices is a square nonsingular matrix. In fact, for an arbitrary invertible matrix T_N of order N the following factorization is possible [68–70]:

$$\begin{aligned} \begin{pmatrix} T_N & 0 \\ 0 & T_N^{-1} \end{pmatrix} &= \begin{pmatrix} -I_N & 0 \\ T_N^{-1} & I_N \end{pmatrix} \begin{pmatrix} I_N & -T_N \\ 0 & I_N \end{pmatrix} \begin{pmatrix} 0 & I_N \\ I_N & T_N^{-1} \end{pmatrix} \\ &= \begin{pmatrix} -I_N & 0 \\ T_N^{-1} & I_N \end{pmatrix} \begin{pmatrix} I_N & -T_N \\ 0 & I_N \end{pmatrix} \begin{pmatrix} I_N & 0 \\ T_N^{-1} & I_N \end{pmatrix} J_{2N}, \end{aligned} \tag{5.193}$$

and the MDL computational structure is defined exactly by the factorization of quasi-diagonal matrix into the product of lower and upper quasi-triangular matrices. An alternative factorization of a quasi-diagonal matrix with opposite main diagonal is defined as [71]

$$\begin{pmatrix} 0 & T_N \\ T_N^{-1} & 0 \end{pmatrix} = \begin{pmatrix} I_N & 0 \\ -T_N^{-1} & I_N \end{pmatrix} \begin{pmatrix} -I_N & T_N \\ 0 & I_N \end{pmatrix} \begin{pmatrix} I_N & 0 \\ T_N^{-1} & I_N \end{pmatrix}. \tag{5.194}$$

A reversible integer-to-integer mapping is simply constructed by the implementation of matrix–vector multiplications in floating-point arithmetic with the results rounded to the nearest integer. If the integer input vector is applied to a matrix on the right-hand side of (5.193), the first half of integer values is processed by the matrix T_N^{-1} (T_N) and then rounded to integer values before adding to the second half of integer values. Hence, the large parts

of the transforms are computed without rounding operations, only the results of matrix–vector multiplications are rounded and added. A different factorization of the general block transform matrix into the product of lower and upper quasi-triangular matrices and quasi-diagonal matrix has been proposed in Refs. [72, 73].

If the factorization (5.193) is applied to the DCT-IV matrix C_N^{IV} and taking into account that $[C_N^{\mathrm{IV}}]^{-1} = C_N^{\mathrm{IV}}$, it becomes [68, 69]:

$$\begin{pmatrix} C_N^{\mathrm{IV}} & 0 \\ 0 & C_N^{\mathrm{IV}} \end{pmatrix} = \begin{pmatrix} -I_N & 0 \\ C_N^{\mathrm{IV}} & I_N \end{pmatrix} \begin{pmatrix} I_N & -C_N^{\mathrm{IV}} \\ 0 & I_N \end{pmatrix} \begin{pmatrix} I_N & 0 \\ C_N^{\mathrm{IV}} & I_N \end{pmatrix} J_{2N}. \tag{5.195}$$

The MDL computational structure defined by (5.193) can be used for invertible integer approximations of the T_N or an invertible approximation of certain scaling operations. Moreover, substituting the factorization (5.193) into a recursive sparse matrix factorization of the transform matrix (e.g., C_N^{IV}) results in further reducing the overall computational complexity compared to noninteger implementation of the transform.

5.6.3.1 Error bound estimates between the exact (scaled) DCT-II and corresponding integer (scaled) DCT-II obtained by dyadic approximation

In the construction of integer DCTs, frequently the floating-point multipliers are either approximated by dyadic rationals or are rounded to the nearest integer. Both dyadic approximation and the rounding procedure introduce truncation errors. In Section 5.5.2 the construction of invertible integer 8-point DCT-II by rounding procedure has been discussed and explicit estimates of truncation errors for integer approximation of ULU factorized rotation matrix $R_2(\varphi) = G_{-\varphi}$ have been specified in Theorem 5.4. On the other hand, in Section 5.4.4 the fast multiplierless 8-point BinDCTs-II and IntDCTs-II have been derived from the rotation-based sparse matrix factorization of DCT-II matrix where multipliers in LUL (ULU) factorized Givens–Jacobi rotation matrices G_φ and $G_{-\varphi}$ were approximated by dyadic rationals. The approximation accuracy between resulting BinDCT-II and IntDCT-II and exact real-valued DCT-II was evaluated by the MSE error between approximated and original DCT-II matrix. However, a little attention has been paid to the analysis of errors caused by dyadic approximation. Recently, the integer-to-integer 8-point DCT-II derived from the orthogonal rotation-based factorization of the matrix C_8^{II} [59, 60] with dyadic approximation of ULU factorized rotation matrix $R_2(\varphi) = G_{-\varphi}$ has been proposed in Ref. [74]. Moreover, the explicit estimates of truncation errors have been derived as follows.

For simplicity, let us use the notation used in Section 5.5.2. Replacing the trigonometric values $\tan\frac{\varphi}{2}$ and $\sin\varphi$ by the dyadic rationals $a = \frac{\beta_a}{2^n}$ and $b = \frac{\beta_b}{2^n}$, $\beta_a, \beta_b, n \in N$, respectively, in the ULU factorization of $G_{-\varphi} = R_2(\varphi)$ given by (5.161), and multiplying the factored matrices we obtain the approximation matrix $\tilde{R}_2(\varphi)$ in the form:

$$\tilde{R}_2(\varphi) = \begin{pmatrix} 1-ab & a+a(1-ab) \\ -b & 1-ab \end{pmatrix}. \tag{5.196}$$

The following theorem states how to estimate truncation errors caused by the dyadic approximation of ULU factorized rotation matrix $R_2(\varphi) = G_{-\varphi}$ [74].

Theorem 5.8: (The integer approximation of rotation matrix $R_2(\varphi)$ represented by ULU structure, where the multipliers are approximated by dyadic rationals and explicit truncation error estimates)

Let $R_2(\varphi)$ be a rotation matrix represented by ULU structure and $\tilde{R}_2(\varphi)$ its approximation matrix with $\varphi \in (0, \frac{\pi}{2}\rangle$. Further, let $a = a(\varphi) = \frac{\beta_a}{2^n} \geq 0$ and $b = b(\varphi) = \frac{\beta_b}{2^n} \geq 0$, $\beta_a, \beta_b, n \in N$ be given with

$$\left|\tan\frac{\varphi}{2} - a\right| \leq 2^{-j} \quad \text{and} \quad |\sin\varphi - b| \leq 2^{-j}$$

for some fixed $j \in N$. Then for arbitrary $\mathbf{x} = [x_0, x_1]^{\mathrm{T}} \in (-2^k, 2^k\rangle \cap Z^2$, a suitable integer approximation $\mathbf{y} = [y_0, y_1]^{\mathrm{T}} \in Z^2$ of $\hat{\mathbf{y}} = \tilde{R}_2\mathbf{x}$ is given by $y_0 = z_2$, $y_1 = z_1$, where

$$z_0 = x_0 + \mathrm{round}(x_1\, a), \quad z_1 = x_1 + \mathrm{round}(-z_0\, b), \quad z_2 = z_0 + \mathrm{round}(z_1\, a).$$

The procedure is left-invertible and its left inverse is $x_0 = w_2$, $x_1 = w_1$, where

$$w_0 = y_0 - \mathrm{round}(y_1\, a), \quad w_1 = y_1 - \mathrm{round}(-w_0\, b), \quad w_2 = w_0 - \mathrm{round}(w_1\, a).$$

Further, the componentwise truncation error can be estimated by

$$|\hat{y}_0 - y_0| \leq (2 + a + a^2 + \sin\varphi\Big(1 + a + \tan\frac{\varphi}{2}\Big)\, 2^{k-j} + \frac{1}{2}(2 + a - ab),$$
$$|\hat{y}_1 - y_1| \leq (1 + a + \sin\varphi)2^{k-j} + \frac{1}{2}(b + 1). \tag{5.197}$$

•

The formulae for y_0, y_1 and x_0, x_1 (after inverse transform) directly follow by applying the input vector $\mathbf{x}$ to three matrices in (5.161). The elegant proof of truncation error estimates (5.197) can be found in Ref. [74]. Explicit overall error bound estimates for two integer-to-integer 8-point DCT-II algorithms (the invertible integer DCT-II described in Section 5.5.2 and IntDCT-II described in Section 5.4.4) with dyadic approximations are also discussed. Provided the integer input vector $\mathbf{x}$ is quantized into 8-bit representation, from the analysis it follows that the overall error bound estimates depend crucially on the range of input data (value of k) if intermediate results rapidly increase through the computation, and approximation quality.

5.7 Summary

Integer transforms are current modern transform technologies especially suitable for low-cost, low-powered and computationally efficient transform-based lossless coding. The resulting integer transforms are comparable to the corresponding original real-valued transforms, and preserve all their mathematical properties (linearity, orthogonality,

orthonormality, symmetry of the basis vectors and recursivity) and performance measures. The methods of integer approximation enable us to construct and flexibly generate the family of integer transforms with arbitrary accuracy and performance with the fast, efficient in-place or even multiplierless implementation using only binary additions and shifts.

Various methods to the construction of integer discrete cosine/sine transforms have been presented. At first the review of basic material from linear algebra, theory of matrices and matrix computations (determinant, orthonormal and orthogonal matrices, triangular matrices, absolute value of a matrix, matrix/vector norms, elementary rotation matrices, elementary transformations, *QR*, *LU* and *PLUS* matrix factorizations) has been presented (Section 5.2). In particular, it is shown how the basic material from linear algebra and matrix theory can be used to derive the factorizations of Givens–Jacobi rotations and Householder reflections into the products of Gauss–Jordan elementary matrices. In order to evaluate the approximation error between the approximated and original transform matrix and to measure the performance of resulting approximated integer transform used in data compression applications, some theoretical criteria are defined (Section 5.3). Finally, in the last three sections (Sections 5.4 and 5.5) various developed methods and design approaches to integer approximation of DCTs and DSTs are described in detail including recent important developments (Section 5.6). The fast integer DCTs and DSTs in the form of generalized signal flow graphs are ready to be used in practical applications.

Problems and Exercises

1. In Section 5.4.1 an integer approximation of the 8-point DCT-II computed via WHT, called the CMT, is described in detail. For the new CMT denoted by CMT_8 (a,b,c,d,e,f,g,h,i,j,k) two 6-bit integer solutions have been obtained. Write computer program to find complete 7- and 8-bit integer solutions satisfying orthogonal and orthonormal conditions given by (5.44)–(5.46) under constraints (5.47)–(5.50). At first, replace inequality (5.49) by (5.51), and then (5.48) by (5.52). For each integer solution derive the corresponding conversion matrix $\bar{T}_8$ with associated CMT matrix C_8^{CMT}. Compute MSE approximation error, performance measures and compare the resulting CMTs to the 8-point DCT-II. Finally, analyze the computational complexity in terms of integer multiplications/additions and multiply-free implementation in the form of additions/shifts.

2. Write computer program for the fast CMT_8 (a,b,c,d,e,f,g,h,i,j,k) implementation shown in Fig. 5.6 and verify its correctness.

3. The open problem in construction of the fast CMT_8 (a,b,c,d,e,f,g,h,i,j,k) is the existence of a sparse matrix factorization of the block matrix $\bar{U}_4$ in the integer domain. Try to find a such factorization. If there exists then modify the fast CMT_8 implementation in Fig. 5.6.

4. Modify the fast CMT_8 (a,b,c,d,e,f,g,h,i,j,k) implementation in Fig. 5.6 for the integer 8-point DST-II computed via WHT and verify its correctness by computer program.

5. Implement the integer 8×8 DCT-II computation based on the fast CMT_8 shown in Fig. 5.6. You note that the normalization factors are reduced to shift

operations. Analyze the total computational complexity for given integer solutions $\{a, b, c, d, e, f, g, h, i, j, k\}$.

6. Derive the orthogonality and orthonormality conditions and constraints on variables to construct the CMT for the integer approximation of 16-point DCT-II computed via WHT. Remember that the conversion matrix T_{16} can be generated recursively, i.e., it is sufficient to derive the explicit and numerical form of the block matrix U_8 and to find its sparse matrix factorization. At first, write computer program to find complete 8-bit integer solutions and compare resulting q integer CMTs with those of presented in Refs. [23 24]. Then design the fast CMT_{16} implementation and verify its correctness by computer program.

7. Following the method to integer approximation of the 8-point DCT-II computed via WHT construct the CMT for SCT defined by (4.9). For $N = 4$ and 8 write computer program to find complete 8-bit integer solutions. For each solution derive the corresponding conversion matrix with associated CMT matrix. Design the fast CMT implementation based on the EOT factorization of SCT matrix and verify its correctness by computer program. Note that the SCT matrix has no recursive property and the corresponding conversion matrix has a block-diagonal structure and consists of two block matrices of the same order.

8. Similarly, following the method to integer approximation of the 8-point DCT-II computed via WHT construct the CMT for SST defined by (4.10). For $N = 4$ and 8 write computer program to find complete 8-bit integer solutions. For each solution derive the corresponding conversion matrix with associated CMT matrix. Design the fast CMT implementation based on the EOT factorization of SST matrix and verify its correctness by computer program. Note that the SST matrix has no recursive property and the corresponding conversion matrix has a block-diagonal structure and consists of two block matrices of the same order.

9. In Section 5.4.2 an approximation method to construct integer DCTs/DSTs is described. In construction of the integer 8-point DCT-II denoted by ICT_8-II (a, b, c, d, e, f, g), the set of seven variables has been used. However, using the relations between cosine/sine elements of C_8^{II} (actually of the matrix C_4^{IV}) given by

$$\cos\frac{3\pi}{16} = \frac{\sqrt{2}}{2}\left(\cos\frac{\pi}{16} + \sin\frac{\pi}{16}\right), \quad \sin\frac{3\pi}{16} = \frac{\sqrt{2}}{2}\left(\cos\frac{\pi}{16} - \sin\frac{\pi}{16}\right),$$

the explicit form of the matrix C_8^{II} can be represented only by five different elements so reducing the set of variables from 7 to 5. Following the procedure in Section 5.4.2 generate by computer program the family of integer ICTs_8-II (a, b, c, d, e) with simplifications and improvements using the 8-bit representation. Design the fast ICT_8-II (a, b, c, d, e) implementation and verify its correctness by computer program. Compute MSE approximation error, performance measures and compare the resulting integer ICTs_8-II (a, b, c, d, e) to the 8-point DCT-II. Compare the computational complexity of integer ICTs_8-II (a, b, c, d, e) with those of ICTs_8-II (a, b, c, d, e, f, g) presented in Section 5.5.2.

10. Similarly, using the relations between cosine/sine elements of C_8^{IV} given by

$$\cos\frac{\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{7\pi}{32}+\sin\frac{7\pi}{32}\right),\quad \sin\frac{\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{7\pi}{32}-\sin\frac{7\pi}{32}\right),$$
$$\cos\frac{3\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{5\pi}{32}+\sin\frac{5\pi}{32}\right),\quad \sin\frac{3\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{5\pi}{32}-\sin\frac{5\pi}{32}\right),$$
$$\cos\frac{5\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{3\pi}{32}+\sin\frac{3\pi}{32}\right),\quad \sin\frac{5\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{3\pi}{32}-\sin\frac{3\pi}{32}\right),$$
$$\cos\frac{7\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{\pi}{32}+\sin\frac{\pi}{32}\right),\quad \sin\frac{7\pi}{32}=\frac{\sqrt{2}}{2}\left(\cos\frac{\pi}{32}-\sin\frac{\pi}{32}\right),$$

and trigonometric identities

$$\cos\frac{\pi}{8}+\sin\frac{\pi}{8}=\sqrt{2}\cos\frac{\pi}{8},\quad \cos\frac{\pi}{8}-\sin\frac{\pi}{8}=\sqrt{2}\sin\frac{\pi}{8},$$
$$\cos\left(\frac{\pi}{8}\pm\alpha\right)=\cos\frac{\pi}{8}\cos\alpha\mp\sin\frac{\pi}{8}\sin\alpha,$$
$$\sin\left(\frac{\pi}{8}\pm\alpha\right)=\sin\frac{\pi}{8}\cos\alpha\pm\cos\frac{\pi}{8}\sin\alpha,$$

where $\alpha=\frac{\pi}{32},\frac{3\pi}{32},\frac{5\pi}{32},\frac{7\pi}{32}$, the explicit form of C_8^{IV} can be represented only by four different elements so reducing the set of variables from 8 to 4. Following the procedure to construct integer $\frac{1}{\sqrt{q}}$ ICT$_8$-IV described in Section 5.4.2 find by computer program complete 8-bit solutions for integer $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (a,b,c,d). Design the fast $\frac{1}{\sqrt{q}}$ ICT$_8$-IV (a,b,c,d) implementation using a suitable fast 8-point DCT-IV algorithm from Section 4.4.4. Verify its correctness by computer program.

11. Modify the fast integer ICT$_8$-II (a,b,c,d,e,f,g) implementation shown in Fig. 5.7 for the fast integer IST$_8$-II (a,b,c,d,e,f,g).

12. Implement the integer 8×8 DCT-II computation based on the fast ICT$_8$-II (a,b,c,d,e,f,g) shown in Fig. 5.7. Analyze the total computational complexity for given integer solutions $\{a,b,c,d,e,f,g\}$.

13. Design the fast integer ICT$_{16}$-II implementation using a suitable fast 16-point DCT-II algorithm (see Sections 4.4.3 and 4.4.4). Remember that the matrix C_{16}^{IV} is recursive. Therefore, you need only derive the fast integer $\frac{1}{\sqrt{q}}$ ICT$_8$-IV.

14. Write computer program to find complete 6-, 7- and 8-bit integer solutions for the integer $\frac{1}{\sqrt{q}}$ ISCTs$_8$ (a,b,c,d,e). Design the fast $\frac{1}{\sqrt{q}}$ ISCT$_8$ (a,b,c,d,e) implementation and verify its correctness by computer program.

15. Write computer program to find complete 6-, 7- and 8-bit integer solutions for the integer $\frac{1}{\sqrt{q}}$ ISSTs$_8$ (a,b,c,d). Design the fast $\frac{1}{\sqrt{q}}$ ISST$_8$ (a,b,c,d) implementation and verify its correctness by computer program.

16. In Section 5.4.3 an orthogonal/orthonormal parametrized GCT as a rationalized approximation of the DCT-II for $N = 8$ is described. For the 8-point DCT-II there are eight different equivalent parametrized forms of the matrix C_8^{II} and only two forms are presented in Section 5.4.3. Derive remaining six forms of the parametrized C_8^{II} matrix.

17. Verify by computer program the fast GCT_8-II (a, b, c, r) implementations shown in Figs. 5.8 and 5.9. Choose suitable approximations of the parameters a, b, c, r by dyadic rationals.

18. Design the fast GCT_8-II (a, b, c, r) implementations for the remaining six equivalent forms of the parametrized C_8^{II} matrix and verify their correctness by computer program. Choose suitable approximations of the parameters a, b, c, r by dyadic rationals.

19. Implement the integer 8×8 DCT-II computation based on the fast GCT_8-II (a, b, c, r) transform for given approximations of the parameters a, b, c, r by dyadic rationals, analyze its total computational complexity and compare with that of presented in Refs. [39–42].

20. Parametrize the DCT-IV matrix C_8^{IV} and construct the parametrized GCT_8-IV. At first, rewrite the explicit form of the matrix C_8^{IV} using the relations between cosine/sine elements (see Problem 10). Design the fast GCT_8-IV implementation and choose suitable approximations of the parameters by dyadic rationals. Verify its correctness by computer program. You need find the sparse matrix factorization of C_8^{IV}.

21. Construct the parametrized GCT_{16} and design the corresponding fast GCT_{16} implementation. Verify its correctness by computer program.

22. Verify by computer program the fast GCMT_8 (a, b, c) implementation shown in Fig. 5.10. Choose suitable approximations of the parameters a, b, c by dyadic rationals.

23. Implement the integer 8×8 DCT-II computation based on the fast GCMT_8-II (a, b, c) for given approximations of the parameters a, b, c by dyadic rationals, analyze its total computational complexity and compare with that of based on GCT_8-II (a, b, c, r).

24. In Section 5.4.4 the constructions of fast multiplierless approximation of the 8-point DCT-II, called BinDCTs-II and IntDCTs-II, are described. In Fig. 5.17(a) and (b) the general forms of the forward and inverse 8-point fast multiplierless BinDCTs-IIC are shown. Verify their correctness by computer program and choose suitable dyadic approximations of multipliers in terms of minimum-adder representation.

25. Implement the forward and inverse 8×8 fast multiplierless approximation of the DCT-II based on BinDCT-IIC shown in Fig. 5.17. Analyze the total computational complexity for given dyadic approximations of multipliers in terms of minimum-adder representation.

26. Verify by computer program the general form of 8-point fast multiplierless BinDCT-IIL shown in Fig. 5.19. Choose suitable dyadic approximations of multipliers in terms of minimum-adder representation.

27. Implement the forward and inverse 8×8 fast multiplierless approximation of the scaled DCT-II based on BinDCT-IIL shown in Fig. 5.19. Analyze the total computational complexity for given dyadic approximations of multipliers in terms of minimum-adder representation.

28. Verify by computer program the general form of 8-point fast multiplierless BinDCT-IIS shown in Fig. 5.21. Choose suitable dyadic approximations of multipliers in terms of minimum-adder representation.

29. Implement the forward and inverse 8×8 fast multiplierless approximation of the DCT-II based on BinDCT-IIS shown in Fig. 5.21. Analyze the total computational complexity for given dyadic approximations of multipliers in terms of minimum-adder representation.

30. Verify by computer program the general form of 8-point fast multiplierless IntDCT-II shown in Fig. 5.23. Choose suitable dyadic approximations of multipliers in terms of minimum-adder representation.

31. Implement the forward and inverse 8×8 fast multiplierless approximation of the DCT-II based on IntDCT-II shown in Fig. 5.23. Analyze the total computational complexity for given dyadic approximations of multipliers in terms of minimum-adder representation.

32. Modify the 8-point fast multiplierless BinDCTs-II and IntDCTs-II for BinDST-II and IntDST-II, respectively.

33. Verify by computer program the general form of 8-point fast multiplierless BinDCT-IV shown in Fig. 5.27. Choose suitable dyadic approximations of multipliers in terms of minimum-adder representation.

34. Modify the 8-point fast multiplierless BinDCT-IV for BinDST-IV.

35. Construct the 8-point fast multiplierless BinDCT-II based on the orthogonal recursive sparse matrix factorization of C_8^{II} given by (4.53) and (4.55). Design the general form of fast multiplierless BinDCT-II and verify its correctness by computer program. You note that trivial butterflies are orthogonal 2×2 rotation matrices. Therefore, they can be also realized by LUL (ULU) structures.

36. Method of the construction of fast multiplierless BinDCTs and IntDCTs presented in Section 5.4.4 offers the high versatility to construct various configurations of BinDCTs and IntDCTs. For example, all rotations at the end of a signal flow graph realized by LUL (ULU) structure can be replaced by DLU structure incorporating scaling factors into the normalization. Construct such configurations.

37. Construct the fast multiplierless BinDCTs-II and IntDCTs-II for $N = 16$ using the (orthogonal) recursive sparse matrix factorizations of DCT-II matrix C_{16}^{II} (see Section 4.4.3.1).

38. Following the method described in Section 5.4.4 construct the fast multiplierless approximations of the DCT-I, DST-I, SCT and SST for $N = 4$ and 8.

39. Dyadic approximations of multipliers in LUL (ULU) and DLU (ULD) structures can be replaced by *round*, *floor* or *ceil* operators. Implement the fast BinDCTs and IntDCTs introducing *round*, *floor* or *ceil* operators.

40. In Section 5.5.1 the lossless 8-point DCTs-II based on *QR*, *LU* and *PLUS* factorizations of the corresponding DCT-II matrices are discussed. Since all DCT and DST matrices for $N > 2$ except for the DST-I are eigenorthonormal, i.e., their determinants

are equal to $+1$, they must have *QR*, *LU* and *PLUS* factorizations. Derive *QR* factorizations both by numerical and analytical procedure for remaining DCT and DST matrices for $N = 8$. To simplify the problem utilize a (recursive) sparse matrix factorization of corresponding transform matrix. Design the *QR*-based DCT/DST computational structure, construct lossless integer LDCT/LDST by dyadic approximations or rounding procedure. Finally, implement the resulting lossless *QR*-based integer DCT/DST.

41. Derive *LU* factorizations for remaining DCT and DST matrices for $N = 8$. Again, to simplify the problem utilize a (recursive) sparse matrix factorization of corresponding transform matrix. Design the *LU*-based DCT/DST computational structure, construct lossless integer LDCT/LDST by rounding procedure. Finally, implement the resulting lossless *LU*-based integer DCT/DST.

42. Derive *PLUS* factorizations for remaining DCT and DST matrices for $N = 8$. To simplify the problem utilize a (recursive) sparse matrix factorization of corresponding transform matrix. Design the *PLUS*-based DCT/DST computational structure, construct lossless integer LDCT/LDST by rounding procedure. Finally, implement the resulting lossless *PLUS*-based integer DCT/DST.

43. In Section 5.5.2 the construction of 8-point invertible integer DCT-II is discussed. Following the method construct the 8-point invertible integer DCTs and DSTs for remaining types of the DCT and DST. Design the fast invertible integer DCTs/DSTs and verify their correctness by computer program.

44. The global method to obtain integer-to-integer 8-point DCT-II is very simple. It is sufficient to use any fast DCT-II algorithm implemented in floating-point arithmetic so that resulting transform coefficients are rounded to the nearest integer. Apply the method to other DCTs and DSTs for $N = 8$.

45. In Section 5.5.3 the construction of (normalized) reversible 8-point DCT-II is described. Apply the method to other DCTs and DSTs for $N = 8$. You note that the concept of normalized transform is associated with the orthogonal sparse matrix factorization of the corresponding transform matrix. Consequently, no normalization of transform coefficients is needed.

46. The methods described in Section 5.5 to construct lossless, invertible integer and reversible DCTs/DSTs can be directly extended to $N = 16$. Repeat the problems above for $N = 16$.

References

[1] D. K. Faddeev and V. N. Faddeeva, *Computational Methods of Linear Algebra*, GIFML, Moskow, 1963 (in Russian), English translation: Dover Publications Inc., New York, 1959.

[2] I. S. Berezin and N. P. Zhidkov, *Computational Methods*, Vol. 2, GIFML, Moskow, 1962 (in Russian).

[3] F. R. Gantmacher, *The Theory of Matrices*, 2nd Edition, Nauka, Moskow, 1966 (in Russian), English translation: Vols. 1 and 2, Chelsea, New York, 1959.

[4] B. P. Demidovic and I. A. Maron, *Basics of Numerical Mathematics*, GIFML, Moscow, 1960 (in Russian).

[5] G. H. Golub and C. F. Van Loan, *Matrix Computations*, 3rd Edition, The Johns Hopkins University Press, Baltimore, 1996.

[6] H. S. Malvar, *Signal Processing with Lapped Transforms*, Artech House, Norwood, MA, 1992, Chapters 1 and 2, pp. 1–80.

[7] F. A. M. L. Bruekers and A. W. M. van den Enden, "New networks for perfect inversion and perfect reconstruction", *IEEE Journal of Selected Areas in Communications*, Vol. 10, No. 1, January 1992, pp. 130–137.

[8] I. Daubechies and W. Sweldens, "Factoring wavelet transforms into lifting steps", *The Journal of Fourier Analysis and Applications*, Vol. 4, No. 3, 1998, pp. 247–269.

[9] P. Hao and Q. Shi, "Matrix factorizations for reversible integer mapping", *IEEE Transactions on Signal Processing*, Vol. 49, No. 10, October 2001, pp. 2314–2324.

[10] P. Hao, "Customizable triangular factorizations of matrices", *Linear Algebra and Its Appications*, Vol. 382, May 2004, pp. 135–154.

[11] Y. She and P. Hao, "On the necessity and sufficiency of PLUS factorizations", *Linear Algebra and Its Applications*, Vol. 400, May 2005, pp. 193–202.

[12] T. Toffoli, "Almost every unit matrix is a ULU", *Linear Algebra and Its Applications*, Vol. 259, July 1997, pp. 31–38.

[13] G. Strang, "Every unit matrix is a LULU", *Linear Algebra and Its Applications*, Vol. 265, November 1997, pp. 165–172.

[14] P. Duhamel and H. Hollmann, "Split-radix FFT algorithm", *Electronics Letters*, Vol. 20, No. 1, January 1984, pp. 14–16.

[15] H. V. Sorensen, M. T. Heideman and C. S. Burrus, "On computing the split-radix FFT", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 1, February 1986, pp. 152–156.

[16] P. Duhamel, "Implementation of split-radix FFT algorithms for complex, real, and real-symmetric data", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. ASSP-34, No. 2, April 1986, pp. 285–295.

[17] W. H. Chen, C. H. Smith and S. C. Fralick, "A fast computational algorithm for the discrete cosine transform", *IEEE Transactions on Communications*, Vol. COM-25, September 1977, pp. 1004–1009.

[18] D. Hein and N. Ahmed, "On a real-time Walsh–Hadamard/cosine transform image processor", *IEEE Transactions on Electromagnetic Compatibility*, Vol. EMC-20, No. 3, August 1978, pp. 453–457.

[19] S. Venkataraman, V. R. Kanchan, K. R. Rao and M. Mohanty, "Discrete transforms via the Walsh–Hadamard transform", *Signal Processing*, Vol. 14, No. 4, June 1988, pp. 371–382.

[20] C. Loeffler, A. Ligtenberg and G. S. Moshytz, "Practical fast 1-D DCT algorithms with 11 multiplications", *Proceedings of the IEEE ICASSP'89*, Glasgow, Scotland, May 1989, pp. 988–991.

[21] N. Suehiro and M. Hatori, "Fast algorithms for the discrete Fourier transform and other transforms", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-34, June 1986, pp. 642–644.

[22] H. W. Jones, D. H. Hein and S. C. Knauer, "The Karhunen–Loève, discrete cosine and related transforms obtained via the Hadamard transform", *Proceedings of the International Telemetring Conference*, Los Angeles, CA, November 1978, pp. 87–98.

[23] R. Srinivasan and K. R. Rao, "An approximation to the discrete cosine transform for $N = 16$", *Signal Processing*, Vol. 5, January 1983, pp. 81–85.

[24] H. S. Kwak, R. Srinivasan and K. R. Rao, "C-matrix transform", *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. ASSP-31, October 1983, pp. 1304–1307.

[25] W. K. Cham and Y. T. Chan, "Integer discrete cosine transforms", *Proceedings of the International Symposium on Signal Processing, Theories, Implementations, and Applications (ISSPA'87)*, Brisbane, Australia, August 1987, pp. 674–676.

[26] C. S. Choy, W. K. Cham and L. Lee, "An LSI implementation of integer cosine transform", *Proceedings of the International Conference on Circuits and Communication Systems (ICCS'88)*, Singapore, November 1988, pp. 17.5.1–17.5.5.

[27] C. S. Choy, W. K. Cham and L. Lee, "An integer transform chip using ASIC technology", *Proceedings of the Picture Coding Symposium (PCS'88)*, Torino, Italy, September 1988, pp. 5.5.1–5.5.2.

[28] W. K. Cham, "Development of integer cosine transforms by the principle of dyadic symmetry", *IEE Proceedings*, Vol. 136, No. 4, Pt. I, August 1989, pp. 276–282.

[29] W. K. Cham and F. S. Wu, "On compatibility of order-8 integer cosine transforms and the discrete cosine transform", *Proceedings of the IEEE Region 10 Conference on Computer and Communication Systems*, Hong Kong, September 1990, pp. 447–449.

[30] W. K. Cham and Y. T. Chan, "An order-16 integer cosine transform", *IEEE Transactions on Signal Processing*, Vol. 39, May 1991, pp. 1205–1208.

[31] W. K. Cham and P. C. Yip, "Integer sinusoidal transforms for image processing", *International Journal of Electronics*, Vol. 70, 1991, pp. 1015–1030.

[32] W. K. Cham, C. S. Choy and W. K. Lam, "A 2-D integer cosine transform chip set and its application", *IEEE Transactions on Consumer Electronics*, Vol. 38, May 1992, pp. 43–47.

[33] P. W. Hawkes, Editor, *Advances in Electronics and Electron Physics*, Vol. 88, Academic Press, Boston, 1994, Chapter 1: Integer sinusoidal transforms, pp. 1–61.

[34] T. C. J. Pang, C. S. O. Choy, C. F. Chan and W. K. Cham, "A self-timed ICT chip for image coding", *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 9, September 1999, pp. 856–860.

[35] S. N. Koh, S. J. Huang and K. K. Tang, "Development of order-16 integer transforms", *Signal Processing*, Vol. 24, No. 3, September 1991, pp. 283–289.

[36] K. M. Cheung, F. Pollara and M. Shahshahani, "Integer cosine transform for image compression", *NASA TECH BRIEF*, Vol. 18, No. 4, Item #44 from JPL New Technology Report, NPO-18564, California Institute of Technology, JET Propulsion Laboratory, Pasadena, CA, April 1994.

[37] S. C. Pei and J. J. Ding, "The integer transforms analogous to discrete trigonometric transforms", *IEEE Transactions on Signal Processing*, Vol. 48, No. 12, December 2000, pp. 3345–3364.

[38] A. Marcek and L. Tomasovic, "A modified integer cosine transform with constant self scalar product of the basis vectors", *Applied Signal Processing*, Vol. 2, 1995, pp. 37–47.

[39] J. D. Allen and S. M. Blonstein, "The multiply-free Chen transform – a rational approach to JPEG", *Proceedings of the Picture Coding Symposium (PCS'91)*, Tokyo, Japan, September 1991, pp. 237–240.

[40] M. Boliek, "Real-time discrete cosine transform chip using generalized Chen transform technology", *Electronics Imaging International*, Boston, MA, September 1991, pp. 428–432.

[41] J. Hara, J. D. Allen, S. M. Blonstein and N. Murayama, "Multiply-free DCT", *Proceedings of the Picture Coding Symposium*, Tokyo, Japan, October 1991.

[42] M. Boliek, J. D. Allen, T. Ryu, Y. Sato and J. Hara, "JPEG image compression hardware implementation with extensions for fixed-rate and compressed image editing applications", *Proceedings of SPIE*, Vol. 2187, San Jose, February 1994.

[43] T. D. Tran, "Fast multiplierless approximation of the DCT", *Proceedings of the 33rd Annual Conference on Information Sciences and Systems*, Baltimore, MD, March 1999, pp. 933–938.

[44] T. D. Tran, "A fast multiplierless block transform for image and video compression", *Proceedings of the IEEE International Conference on Image Processing (ICIP'99)*, Kobe, Japan, October 1999, pp. 822–826.

[45] T. D. Tran, "The BinDCT: fast multiplierless approximation of the DCT", *IEEE Signal Processing Letters*, Vol. 7, June 2000, pp. 141–144.

[46] J. Liang and T. D. Tran, "Fast multiplierless approximations of the DCT with the lifting scheme", *IEEE Transactions on Signal Processing*, Vol. 49, No. 12, December 2001, pp. 3032–3044.

[47] Y. J. Chen, S. Oraintara and T. Nguyen, "Integer discrete cosine transform (IntDCT)", *Proceedings of the 2nd International Conference on Information, Communications and Signal Processing*, Singapore, December 1999.

[48] S. Oraintara, Y. J. Chen and T. Nguyen, "Integer fast Fourier transform (IntFFT)", *Proceedings of the IEEE ICASSP'2001*, Salt Lake City, UT, May 2002.

[49] S. Oraintara, Y. J. Chen and T. Nguyen, "Integer fast Fourier transform (IntFFT)", *IEEE Transactions on Signal Processing*, Vol. 50, No. 3, March 2002, pp. 607–618.

[50] Y. J. Chen, S. Oraintara, T. D. Tran, K. Amaratunga and T. Q. Nguen, "Multiplierless approximation of transforms using lifting scheme and coordinate descent with adder constraint", *Proceedings of the IEEE ICASSP'2002*, Vol. 3, Orlando, FL, May 2002, pp. 3136–3139.

[51] Y. J. Chen, S. Oraintara, T. D. Tran, K. Amaratunga and T. Q. Nguyen, "Multiplierless approximation of transforms with adder constraint", *IEEE Signal Processing Letters*, Vol. 9, No. 11, November 2002, pp. 344–347.

[52] S. C. Chan and P. M. Liu, "Multiplier-less discrete sinusoidal and lapped transforms using sum-of-powers-of-two (SOPOT) coefficients", *Proceedings of the International Symposium on Circuits and Systems (ISCAS'2001)*, Vol. 2, Sydney, Australia, May 2001, pp. 13–16.

[53] S. C. Chan and P. M. Liu, "An efficient multiplierless approximation of the Fast Fourier transform using sum-of-powers-of-two (SOPOT) coefficients", *IEEE Signal Processing Letters*, Vol. 9, No. 10, October 2002, pp. 322–325.

[54] L. Z. Cheng, H. Xu and Y. Luo, "Integer discrete cosine transform and its fast algorithm", *Electronics Letters*, Vol. 37, No. 1, January 2001, pp. 64–65, Errata: *Electronics Letters*, Vol. 37, No. 12, July 2001, p. 803.

[55] Y. Zeng, G. Bi and Z. Lin, "Integer sinusoidal transforms based on lifting factorization", *Proceedings of the IEEE ICASSP'2001*, Salt Lake City, UT, May 2001, pp. 1181–1184.

[56] Y. Zeng, L. C. Cheng, G. Bi and A. C. Kot, "Integer DCTs and fast algorithms", *IEEE Transactions on Signal Processing*, Vol. 49, No. 11, November 2001, pp. 2774–2782.

[57] Y. Zeng, G. Bi and Z. Lin, "Lifting factorization of discrete W transform", *Circuits Systems and Signal Processing*, Vol. 21, No. 3, May–June 2002, pp. 277–298.

[58] W. Philips, "The lossless DCT for combined lossy/lossless image coding", *Proceedings of the IEEE International Conference on Image Processing (ICIP'98)*, Vol. 3, Chicago, IL, 1998, pp. 871–875.

[59] G. Plonka and M. Tasche, "Invertible integer DCT algorithms", *Applied and Computational Harmonic Analysis*, Vol. 15, No. 1, July 2003, pp. 70–88.

[60] G. Plonka and M. Tasche, "Integer DCT-II by lifting steps". In: W. Haussmann, K. Jetter, M. Reimer and J. Stokler (Editors), *International Series in Numerical Mathematics*, Vol. 145, Birkhauser, Basel, 2003, pp. 235–252.

[61] G. Plonka, "A global method for invertible integer DCT and integer wavelet algorithms", *Applied and Computational Harmonic Analysis*, Vol. 16, No. 2, March 2004, pp. 79–110.

[62] K. Komatsu and K. Sezaki, "Reversible discrete cosine transform", *Proceedings of the IEEE ICASSP'98*, Seattle, WA, May 1998, pp. 1769–1772.

[63] K. Komatsu and K. Sezaki, "Design of lossless block transforms and filter banks for image coding", *IEICE Transactions Fundamentals*, Vol. E82-A, No. 8, August 1999, pp. 1656–1664.

[64] K. Komatsu and K. Sezaki, "Design of lossless LOT and its performance evaluation", *Proceedings of the IEEE ICASSP'2000*, Istanbul, Turkey, June 2000, pp. 2119–2122.

[65] T. I. Haweel, "A new square wave transform based on DCT", *Signal Processing*, Vol. 81, No. 11, Nov. 2001, pp. 2309–2319.

[66] J. Wang, J. Sun and S. You, "1-D and 2-D transforms from integers to integers", *Proceedings of the IEEE ICASSP'2003*, Vol. 2, Hong Kong, April 2003, pp. 549–552.

[67] R. Geiger and G. Schuller, "Integer low delay and MDCT filter banks", *Proceedings of the 36th Asilomar Conference on Signals, Systems and Computers*, Vol. 1, Pacific Grove, CA, November 2002, pp. 811–815.

[68] R. Geiger, Y. Yokotani and G. Schuller, "Improved integer transforms for lossless audio coding", *Proceedings of the 37th Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, November 2003, pp. 2119–2123.

[69] R. Geiger, Y. Yokotani, G. Schuller and J. Herre, "Improved integer transforms using multi-dimensional lifting", *Proceedings of the IEEE ICASSP'2004*, Montreal, Canada, May 2004, pp. 1005–1008.

[70] Y. Yokotani, S. Oraintara, R. Geiger, G. Schuller and K. R. Rao, "A comparison of integer fast Fourier transforms for lossless coding", *Proceedings of the International Symposium on Communications and Information Technologies 2004 (ISCIT 2004)*, Sapporo, Japan, October 2004, pp. 1069–1073.

[71] H. Huang, R. Yu, X. Lin and S. Rahardja, "Method for realising reversible integer type-IV discrete cosine transform", *Electronics Letters*, Vol. 40, No. 8, April 2004, pp. 514–515.

[72] J. Li, "Reversible FFT and MDCT via matrix lifting", *Proceedings of the IEEE ICASSP'2004*, Vol. 4, Montreal, Canada, May 2004, pp. 173–176.

[73] J. Li, "Low noise reversible MDCT (RMDCT) and its application in progressive-to-lossless embedded audio coding", *IEEE Transactions on Signal Processing*, Vol. 53, No. 5, May 2005, pp. 1870–1880.

[74] M. Primbs, "Worst-case error analysis of lifting-based fast DCT-algorithms", *IEEE Transactions on Signal Processing*, Vol. 53, No. 8, Part 2, August 2005, pp. 3211–3218.

APPENDIX A

A.1 Vector spaces

In this section, some important concepts in vector spaces are reviewed. Some examples are listed to begin this review.

Example 1: In a three-dimensional Euclidean space, any plane containing the origin is a vector space over the field of real numbers.

Example 2: Polynomials of degree n with real coefficients form a vector space.

Example 3: Solutions of a linear n-th-order homogeneous differential equation form a vector space.

Example 4: Solutions of a system of n linear homogeneous equations form a vector space.

The common properties that make these examples, vector spaces can be stated more formally in the following definition.

Definition 1: *Vector space*: A nonempty set V of elements $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, ... called vectors, is a vector space, if the operations of *vector addition* and *scalar multiplication* on the elements are defined as follows:

I. *Vector addition*: For vectors $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, ... in V,
   (a) $\mathbf{a}+\mathbf{b}$ is in V (closure).
   (b) $\mathbf{a}+\mathbf{b}=\mathbf{b}+\mathbf{a}$ (commutativity).
   (c) $(\mathbf{a}+\mathbf{b})+\mathbf{c}=\mathbf{a}+(\mathbf{b}+\mathbf{c})$ (associativity).
   (d) there exists a $\mathbf{0}$ in V such that $\mathbf{0}+\mathbf{a}=\mathbf{a}$ (additive identity).
   (e) there exists a $-\mathbf{a}$ for every $\mathbf{a}$ in V such that $\mathbf{a}+(-\mathbf{a})=\mathbf{0}$ (additive inverse).

II. *Scalar multiplication*: For vectors $\mathbf{a}$, $\mathbf{b}$, ... in V and scalars m, n, ... over a field F
   (a) $m\mathbf{a}$ is in V (closure).
   (b) $m(n\mathbf{a})=(mn)\mathbf{a}$ (associativity).

(c) $m(\mathbf{a}+\mathbf{b})=m\mathbf{a}+m\mathbf{b}$ (distributivity).

(d) $(m+n)\mathbf{a}=m\mathbf{a}+n\mathbf{a}$ (distributivity).

(e) there exists a 1 in F such that $1.\mathbf{a}=\mathbf{a}$ (multiplicative identity).

In particular, when F is the field of real numbers $\boldsymbol{R}$, V is a real vector space, and when F is the field of complex numbers $\boldsymbol{C}$, V is a complex vector space. A more detailed example is given here to illustrate the important aspects of the definition.

Example 5: The solution set S of the linear homogeneous system

$$a_{11}x+a_{12}y=0$$
$$a_{21}x+a_{22}y=0$$

is a vector space over the real numbers $\boldsymbol{R}$ (if a_{ij}'s are in $\boldsymbol{R}$).

To see this, let $\mathbf{u}=(x_1,y_1)^{\mathrm{T}}$ and $\mathbf{v}=(x_2,y_2)^{\mathrm{T}}$ be two such solutions in S. The closure property in the vector addition is clear, because $\mathbf{u}+\mathbf{v}$ and $\mathbf{v}+\mathbf{u}$ are obviously also solutions in S. The trivial solution $\mathbf{0}=(0,0)^{\mathrm{T}}$ satisfies the homogeneous equations. Therefore S has an additive identity. Also, if $\mathbf{u}$ is a solution, so is $-\mathbf{u}$. Thus $\mathbf{u}$ has an additive inverse. The scalar multiplication properties can be demonstrated in a similar way. Hence S is a vector space. We have used the superscript T to denote transposition.

A counter-example is important. Here is one.

Example 6: The set of all three-dimensional vectors over $\boldsymbol{R}$, with non-vanishing magnitudes is not a vector space, since there is no additive identity in the set.

The questions of how the vectors in a vector space are related and of how to represent a vector space lead to the concept of linear independence, which is set out in the following definition.

Definition 2: *Linear independence*: a set of vectors, $\mathbf{a}_i$, $i=1, 2, \ldots, m$ in a vector space V is said to be linearly independent if the linear combination.

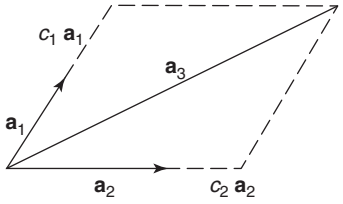
$$\sum_{i=1}^{m} c_i\mathbf{a}_i$$

vanishes ($=\mathbf{0}$) only when all the scalars c_i's are zero. Otherwise, the vectors $\mathbf{a}_i$'s are linearly dependent.

In particular, the set of all possible linear combinations is also a vector space S called the "span" of $\{\mathbf{a}_i,\ i=1, 2, \ldots, m\}$. If the vectors $\mathbf{a}_i$'s are linearly independent, they form a "basis" for the span S. Note that S is called a "subspace" of V if the span of $\{\mathbf{a}_i,\ i=1, 2, \ldots, m\}$ is not V.

Example 7: Two non-collinear nonzero vectors in the Euclidean three-dimensional space are linearly independent and form a basis for the two-dimensional vector subspace, which is the plane containing these two vectors passing through the origin.

Example 8: Any three nonzero vectors which are coplanar are linearly dependent since one can always be expressed as a linear combination of the remaining two as is evident in the following diagram.



i.e., $\mathbf{a}_3 = c_1\mathbf{a}_1 + c_2\mathbf{a}_2$ or $c_1\mathbf{a}_1 + c_2\mathbf{a}_2 - \mathbf{a}_3 = \mathbf{0}$. Noting that the vanishing combination is obtained by nonzero coefficients, we see that the vectors are linearly dependent.

Example 8 is a special case of a much more general result which states that any $(n+1)$ nonzero vectors in an n-dimensional vector space (the span of n linearly independent vectors) are linearly dependent. The result stated more formally in the following theorem touches on a wide range of engineering applications.

Theorem 1: Any nonzero vector $\mathbf{v}$, in an n-dimensional vector space V can be expressed as a linear combination of n linearly independent vectors $\mathbf{a}_i$'s, $i = 1, 2, \ldots, n$, in V, i.e.,

$$\mathbf{v} = \sum_{i=1}^{n} c_i \mathbf{a}_i. \tag{A.1}$$

This is demonstrated in the next example.

Example 9: The vector $(1, 2, 3)^{\mathrm{T}}$ in a three-dimensional vector space can be expressed as a linear combination of the three linearly independent vectors

$$\mathbf{a}_1 = (1,1,1)^{\mathrm{T}}, \quad \mathbf{a}_2 = (0,1,1)^{\mathrm{T}} \quad \text{and} \quad \mathbf{a}_3 = (0,0,1)^{\mathrm{T}} \text{ so that}$$

$$(1,2,3)^{\mathrm{T}} = (1,1,1)^{\mathrm{T}} + (0,1,1)^{\mathrm{T}} + (0,0,1)^{\mathrm{T}} = \mathbf{a}_1 + \mathbf{a}_2 + \mathbf{a}_3.$$

We note here that if the linearly independent vectors are

$$\mathbf{b}_1 = (1,0,0)^{\mathrm{T}}, \quad \mathbf{b}_2 = (0,1,0)^{\mathrm{T}} \quad \text{and} \quad \mathbf{b}_3 = (0,0,1)^{\mathrm{T}}$$

and we will have

$$(1,2,3)^{\mathrm{T}} = \mathbf{b}_1 + 2\mathbf{b}_2 + 3\mathbf{b}_3.$$

This illustrates that there can be more than one basis set of linearly independent vectors for a given vector space. These different basis sets are "equal" in that they all span the same

vector space. But there are some more "equal" than others, because they are easier to use in (A.1). To make this clear, we give the definitions of inner product and orthogonality.

Definition 3: *Inner product*: An inner product on a vector space V is a function that maps a pair of vectors **a**, **b** in V on to a scalar function denoted by $<\mathbf{a}, \mathbf{b}>$ over the field F such that the following will hold for arbitrary vectors **a**, **b**, **c** in V and scalars m, n over F

(a) $<\mathbf{a}, \mathbf{b}> = <\mathbf{b}, \mathbf{a}>$ (commutativity).

(b) $<m\mathbf{a} + n\mathbf{b}, \mathbf{c}> = m<\mathbf{a}, \mathbf{c}> + n<\mathbf{b}, \mathbf{c}>$ (distributivity).

(c) $<\mathbf{a}, \mathbf{a}> > 0$ if $\mathbf{a} > \mathbf{0}$ (positive definiteness).

Example 10: The "dot" product between two physical vectors **a** and **b** defined as

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos\theta,$$

where $|\cdot|$ denotes the magnitude of the vector, and θ is the angle between the two vectors is an inner product as can be easily verified.

Any vector space for which a real inner product function has been defined is called a Euclidean space. Many of the geometrical (metric) properties of the well-known three-dimensional space can be usefully transferred to such a vector space. One such property is that of orthogonality. Note from Example 10 that

$$\mathbf{a} \cdot \mathbf{b} = 0 \quad \text{if } \theta = \frac{\pi}{2},$$

i.e., when **a** is perpendicular to **b**. In a similar way two vectors in a vector space are said to be orthogonal if their inner product vanishes. This is stated in the following definition.

Definition 4: *Orthogonality*: Two nonzero vectors **a** and **b** in a vector space V are said to be orthogonal if their inner product vanishes, i.e.,

$$<\mathbf{a}, \mathbf{b}> = 0. \tag{A.2}$$

Example 11: The vector space of n-tuples has an inner product function defined between two vectors **a** and **b** as

$$<\mathbf{a}, \mathbf{b}> = \sum_{i=1}^{n} a_i b_i, \tag{A.3}$$

where $\{a_i\}$ and $\{b_i\}$ are components of the vectors **a** and **b**, respectively. **a** and **b** are said to be orthogonal to each other if the sum on the righthand side of (A.3) vanishes. In particular, we note, for instance, that the two 4-tuples

$$(1, -1, 1, -1)^{\mathrm{T}} \quad \text{and} \quad (1, 1, 1, 1)^{\mathrm{T}}$$

are orthogonal to each other.

It is simple to visualize in a three-dimensional Euclidean space that any three mutually orthogonal vectors must be linearly independent because it is not possible to express one as a linear combination of the other two. This means that any three mutually orthogonal

vectors in the three-dimensional Euclidean space form a basis and span the vector space. These orthogonal basis vectors are the "more equal" basis vectors, because they are much easier to use in representing other vectors in the vector space. This is shown in the next example.

Example 12: Find the linear combination of **a**, **b** and **c** that will give vector **d** in a three-dimensional vector space V, in which the inner product is defined and **a**, **b**, **c** are orthogonal basis vectors.

Let $\mathbf{d} = d_1\mathbf{a} + d_2\mathbf{b} + d_3\mathbf{c}$. If we form the inner product with **a** on both sides, we get the equation

$$<\mathbf{a}, \mathbf{d}> = d_1 <\mathbf{a}, \mathbf{a}> \quad \text{or} \quad d_1 = <\mathbf{a}, \mathbf{d}>/<\mathbf{a}, \mathbf{a}>,$$

since the inner products of **a** with **b** and **c** are zero. d_2 and d_3 can be just as easily obtained. It can be seen very clearly that such simple solutions are not possible when the basis vectors are not orthogonal.

The above example is a special case of a much more general result which is now stated in the form of a theorem.

Theorem 2: If $\{\mathbf{a}_i,\ i = 1,\ 2, \ldots,\ n\}$ is an orthogonal set of basis vectors for an n-dimensional vector space V, then any nonzero vector **d** in V can be expressed as a linear combination of these basis vectors, i.e.,

$$\mathbf{d} = \sum_{i=1}^{n} d_i \mathbf{a}_i,$$

where the scalar coefficients are given by

$$d_i = <\mathbf{a}_i, \mathbf{d}>/<\mathbf{a}_i, \mathbf{a}_i>. \tag{A.4}$$

If these orthogonal basis vectors are "more equal" than just linearly independent basis vectors, is it possible to make or produce a mutually orthogonal set from a merely linearly independent set? To answer this question, we should really proceed in two stages. First, we should investigate whether there always exists such an orthogonal basis set in a vector space V. If there is, then we ask how to find such a set. The following existence theorem answers the first part of the question, and the well-known Gram–Schmidt's procedure of orthogonalization which is the main part of the constructive proof of this theorem, answers the second part of the question.

Theorem 3: Every finite dimensional Euclidean vector space (one in which an inner product function has been defined) possesses an orthogonal basis.

The proof is outlined here in order to highlight the Gram–Schmidt's procedure.

Let $\{\mathbf{a}_i, i = 1, 2, \ldots, n\}$ be a linearly independent set of basis vectors in the vector space V. We will seek to produce a set of basis vectors $\{\mathbf{b}_i, i = 1, 2, \ldots, n\}$, which is orthogonal. We proceed as follows:

1. Let $\mathbf{b}_1 = \mathbf{a}_1$ (this is an arbitrary choice, any of the other $\mathbf{a}$'s could be chosen).
2. Let $\mathbf{b}_2 = \mathbf{a}_2 + c_{21}\mathbf{b}_1$, in such a way that $<\mathbf{b}_2, \mathbf{b}_1> = 0$. This gives
$$c_{21} = -<\mathbf{a}_2, \mathbf{b}_1>/<\mathbf{b}_1, \mathbf{b}_1>.$$
3. Let $\mathbf{b}_3 = \mathbf{a}_3 + c_{32}\mathbf{b}_2 + c_{31}\mathbf{b}_1$, in such a way that $\mathbf{b}_3$ will be orthogonal to the two vectors, $\mathbf{b}_1$ and $\mathbf{b}_2$ already constructed.

   This means that
$$c_{32} = -<\mathbf{a}_3, \mathbf{b}_2>/<\mathbf{b}_2, \mathbf{b}_2> \quad \text{and} \quad c_{31} = -<\mathbf{a}_3, \mathbf{b}_1>/<\mathbf{b}_1, \mathbf{b}_1>.$$
4. Continuing in this fashion, the general formula for the Gram–Schmidt's procedure is obtained as
$$\mathbf{b}_m = \mathbf{a}_m - \sum_{i=1}^{m-1} \frac{\langle \mathbf{a}_m, \mathbf{b}_i \rangle}{\langle \mathbf{b}_i, \mathbf{b}_i \rangle} \mathbf{b}_i, \quad m = 2, 3, \ldots, n. \tag{A.5}$$

Thus, it is always possible to construct the set $\{\mathbf{b}_i, i = 1, 2, \ldots, n\}$ from the basis $\{\mathbf{a}_i\}$. If in addition, $<\mathbf{b}_i, \mathbf{b}_i> = 1$ for all i, $\{\mathbf{b}_i\}$ is an *orthonormal* basis set for the vector space V. We should note that the set $\{\mathbf{b}_i\}$ is not unique since the basis vectors $\{\mathbf{a}_i\}$ can enter the Gram–Schmidt's procedure in any order. A specific example follows.

Example 13: Given $\mathbf{a}_1 = (1,\ 1,\ 1)^{\mathrm{T}}$, $\mathbf{a}_2 = (0,\ 1,\ 1)^{\mathrm{T}}$ and $\mathbf{a}_3 = (0,\ 0,\ 1)^{\mathrm{T}}$ as a basis in the three-dimensional vector space, find an orthogonal set $\{\mathbf{b}_i,\ i = 1,\ 2,\ 3\}$ using the Gram–Schmidt's process.

Let $\mathbf{b}_3 = \mathbf{a}_3 = (0, 0, 1)^{\mathrm{T}}$, then let $\mathbf{b}_2 = \mathbf{a}_2 + c_{23}\mathbf{b}_3$ such that $<\mathbf{b}_2, \mathbf{b}_3> = 0$. This condition determines the constant c_{23}, so that

$$c_{23} = -<\mathbf{a}_2, \mathbf{b}_3>/<\mathbf{b}_3, \mathbf{b}_3> = -<\mathbf{a}_2, \mathbf{a}_3>/<\mathbf{a}_3, \mathbf{a}_3> = -1,$$

giving

$$\mathbf{b}_2 = \mathbf{a}_2 - \mathbf{a}_3 = (0, 1, 0)^{\mathrm{T}}.$$

Proceed now to the third and last vector by letting

$$\mathbf{b}_1 = \mathbf{a}_1 + c_{12}\mathbf{b}_2 + c_{13}\mathbf{b}_3 \quad \text{and requiring that it be orthogonal}$$

to the two vectors already constructed, i.e., $<\mathbf{b}_1, \mathbf{b}_2> = <\mathbf{b}_1, \mathbf{b}_3> = 0$. From these two conditions, the constants are

$$c_{12} = -<\mathbf{a}_1, \mathbf{b}_2>/<\mathbf{b}_2, \mathbf{b}_2> = -1 \quad \text{and} \quad c_{13} = -<\mathbf{a}_1, \mathbf{b}_3>/<\mathbf{b}_3, \mathbf{b}_3> = -1.$$

Therefore, $\mathbf{b}_1 = (1, 0, 0)^{\mathrm{T}}$. Note, in addition, that the vectors are also normalized with unit magnitudes.

In this last example, if a matrix is constructed using the three row vectors $\mathbf{a}_i^{\mathrm{T}}$, $i = 1, 2, 3$, so that

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_3^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ \mathbf{a}_1^{\mathrm{T}} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix},$$

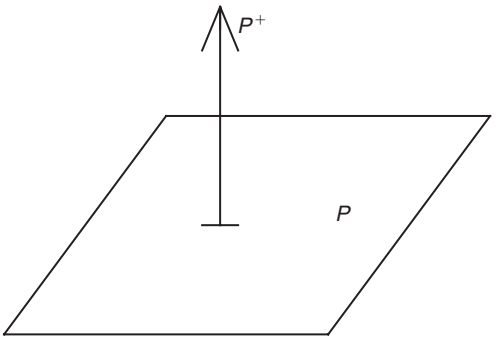
then the Gram–Schmidt's process can be represented by a set of row operations given by the transformation matrix

$$\mathbf{T} = \begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix}, \quad \text{so that } \mathbf{TA} = \begin{bmatrix} \mathbf{b}_3^{\mathrm{T}} \\ \mathbf{b}_2^{\mathrm{T}} \\ \mathbf{b}_1^{\mathrm{T}} \end{bmatrix} = \mathbf{B}.$$

The matrix $\mathbf{B}$, in which the rows are mutually orthogonal is called an orthogonal matrix. Thus the Gram–Schmidt's procedure can be regarded as the transformation that will take a matrix of full row rank (i.e., linearly independent rows) and turn it into an orthogonal matrix (i.e., mutually orthogonal rows).

The Gram–Schmidt's procedure and the concept of orthogonal basis for a vector space combine to produce a very important result. This result states that if U is a subspace of a finite dimensional Euclidean space V, it is always possible to find a subspace U^+, which is orthogonal to U, and together with it to make up the complete V. Before stating this important result as a theorem, we will see its realization in the three-dimensional Euclidean space in the following example.

Example 14: Let V be the three-dimensional Euclidean vector space. Then P, any plane passing through the origin, is a subspace of V as shown in the following diagram.



The normal to the plane through the origin, denoted by P^+, is orthogonal to all vectors lying in P, and P^+ is a subspace of V. Note that choosing any two orthogonal vectors in P emanating from the origin, and combining with a vector along P^+ will produce three mutually orthogonal vectors in V, which form its basis. Thus, P and P^+ together make up the vector space V. P^+ is called the orthogonal complement of P in V.

Now the theorem.

Theorem 4: If U is a subspace of a finite dimensional Euclidean space V, then there exists a unique subspace U^+, orthogonal to U such that $U \oplus U^+ = V$ ($\oplus$ denotes direct sum, i.e., the collection of all vectors in both subspaces). Each of U and U^+ is the orthogonal complement of the other in V.

The proof of this theorem is instructive and highlights the usefulness of the Gram–Schmidt's procedure. The proof is outlined in what follows.

Proof: Let $\{\mathbf{a}_1, \ldots, \mathbf{a}_r\}$ be a basis for U. Then it is not difficult to see that by extending this set, we can find a set

$$\{\mathbf{a}_1, \ldots, \mathbf{a}_r, \mathbf{c}_{r+1}, \ldots, \mathbf{c}_n\},$$

which will be a basis for the vector space V. By applying the Gram–Schmidt's procedure on this set of basis vectors as ordered, an orthogonal set of basis vectors can be found denoted by

$$\{\mathbf{b}_1, \ldots, \mathbf{b}_r, \mathbf{b}_{r+1}, \ldots, \mathbf{b}_n\}.$$

Based on the procedure, $\{\mathbf{b}_1, \ldots, \mathbf{b}_r\}$ are linear combinations of $\{\mathbf{a}_1, \ldots, \mathbf{a}_r\}$ and form an orthogonal basis for U. The remaining members $\{\mathbf{b}_{r+1}, \ldots, \mathbf{b}_n\}$ are orthogonal to all vectors in U, again by the Gram–Schmidt's construction, and span a subspace U^+. That $U \oplus U^+ = V$ is obvious. Now, since every vector which is orthogonal to U must be in U^+, U^+ is unique. Note that although the basis set $\{\mathbf{b}_{r+1}, \ldots, \mathbf{b}_n\}$ is not unique, the subspace U^+ which it spans is.

An example in four dimensions concludes this section.

Example 15: Let $\mathbf{a}_1 = (0,\ 0,\ 0,\ 1)^{\mathrm{T}}$, $\mathbf{a}_2 = (0,\ 0,\ 1,\ 1)^{\mathrm{T}}$, $\mathbf{a}_3 = (0,\ 1,\ 1,\ 1)^{\mathrm{T}}$ and $\mathbf{a}_4 = (1,\ 1,\ 1,\ 1)^{\mathrm{T}}$ be the basis of a four-dimensional Euclidean space V. Let us choose $\{\mathbf{a}_1, \mathbf{a}_2\}$ as the basis for the subspace U and try to find U^+, its orthogonal complement in V.

Let $\mathbf{b}_1 = \mathbf{a}_1$, then $\mathbf{b}_2 = \mathbf{a}_2 - \mathbf{b}_1 <\mathbf{a}_2, \mathbf{b}_1> / <\mathbf{b}_1, \mathbf{b}_1> = (0,\ 0,\ 1,\ 0)^{\mathrm{T}}$. Thus, $\{\mathbf{b}_1, \mathbf{b}_2\}$ form an orthogonal basis for U. We now continue this process outside of U.

$$\text{Let } \mathbf{b}_3 = \mathbf{a}_3 - \sum_{i=1}^{2} \mathbf{b}_i <\mathbf{a}_3, \mathbf{b}_i> / <\mathbf{b}_i, \mathbf{b}_i> = (0, 1, 0, 0)^{\mathrm{T}}.$$

$$\text{Similarly, } \mathbf{b}_4 = \mathbf{a}_4 - \sum_{i=1}^{3} \mathbf{b}_i <\mathbf{a}_4, \mathbf{b}_i> / <\mathbf{b}_i, \mathbf{b}_i> = (1, 0, 0, 0)^{\mathrm{T}}.$$

Note that $\{\mathbf{b}_3, \mathbf{b}_4\}$ spans a subspace where all vectors are orthogonal to the subspace U. This is the orthogonal complement U^+ that we were looking for.

Problems and Exercises A.1

1. Determine whether each of the following statement is true or false:
   (a) The set of all $m \times n$ matrices defined over the field of real numbers $\boldsymbol{R}$ is a vector space.
   (b) The set of all n-th degree polynomials defined over $\boldsymbol{R}$ with real coefficients is a vector space.
   (c) Any plane in a three-dimensional Euclidean space is a vector subspace.
   (d) The set of all possible sinusoidal signals limited within a given frequency band is a vector space.
   (e) The set of all non-vanishing dyadic signals limited in a given sequency band is a vector space.
2. Determine whether each set of the following vectors is linearly independent:
   (a) $\mathbf{a}_1 = (1, 1, 1)^{\mathrm{T}}$, $\mathbf{a}_2 = (1, 2, 2)^{\mathrm{T}}$, $\mathbf{a}_3 = (1, 2, 3)^{\mathrm{T}}$.
   (b) $\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix}$.
   (c) $\mathbf{a}_1 = 1$, $\mathbf{a}_2 = x$, $\mathbf{a}_3 = x^2$, $\mathbf{a}_4 = x^3$, for x in $(-1, 1)$.
   (d) $\mathbf{a}_1 = \sin x$, $\mathbf{a}_2 = \cos x$, $\mathbf{a}_3 = \sin 2x$, $\mathbf{a}_4 = \sin x \cos x$, x in $(0, 2\pi)$.
   (e) (1, 1, 1), (1, 2, 2), (0, 0, 0).
3. Prove the inequality: $<\mathbf{a} + \mathbf{b}, \mathbf{a} + \mathbf{b}> \leq <\mathbf{a}, \mathbf{a}> + <\mathbf{b}, \mathbf{b}>$.
4. Find the linear combination of the matrices in Exercise 2(b) which will produce the matrix
$$\begin{bmatrix} 3 & 4 \\ 5 & 6 \end{bmatrix}.$$
5. If the inner product between two matrices is defined as
$$<\mathbf{A}, \mathbf{B}> = \sum_i \sum_j a_{ij} b_{ij},$$
determine which pairs of matrices in Exercise 2(b) are orthogonal.
6. Based on Exercise 5 and the Gram–Schmidt's procedure, generate an orthogonal basis for the vector space of 2×2 matrices.
7. For the functions: $f_0 = 1, f_1 = x, f_2 = x^2, f_3 = x^3$, with x in $(-1, +1)$, define an inner product
$$<f_i, f_j> = \int_{-1}^{1} f_i(x) f_j(x) \mathrm{d}x,$$
   (a) show that the given functions are not mutually orthogonal,
   (b) use the Gram–Schmidt's procedure to generate an orthogonal basis.

8. If V is the vector space of all polynomials defined over the real line $\boldsymbol{R}$ with degree less than or equal to three, and W is the vector space of all polynomials defined over $\boldsymbol{R}$ of degree less than or equal to one,

   (a) show that W is a subspace of V,

   (b) find the orthogonal complement W^+ of W in V.

9. Assume an orthonormal basis $\{\mathbf{a}_1, \ldots, \mathbf{a}_n\}$ for an n-dimensional Euclidean space V:

   (a) Prove the Bessel's inequality for any vector $\mathbf{v}$ in V

$$\sum_{i=1}^{m} <\mathbf{v}, \mathbf{a}_i>^2 \leq <\mathbf{v}, \mathbf{v}>, \quad \text{for } m < n.$$

   (b) Prove the Parseval's identity for $\mathbf{u}$, $\mathbf{v}$ in V

$$\sum_{i=1}^{n} <\mathbf{u}, \mathbf{a}_i><\mathbf{v}, \mathbf{a}_i> = <\mathbf{u}, \mathbf{v}>.$$

10. Let U be the vector subspace spanned by the vectors

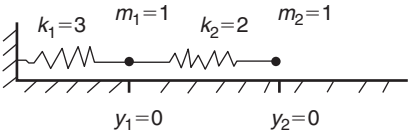
$$(1, 1, 0, 1) \text{ and } (0, 1, 1, 0),$$

    in a four-dimensional vector space V. Show that U^+, its orthogonal complement in V, is the solution space of two homogeneous linear equations in four unknowns. (**Hint**: any member of U^+ must be orthogonal to any vector in U which is a linear combination of the two given vectors.)

A.2 The matrix eigenvalue problem

As we have seen in examples in the foregoing section, vector spaces and homogeneous linear simultaneous equations are closely related. In particular, the so-called "eigenvalue" problem is a classic example of such a system. A very simple example leads off our discussion.

Example 16: The vibrations of a coupled spring–mass system as represented by the following diagram (ignoring all gravitational forces and frictions)



where m_1, m_2, k_1, k_2, y_1 and y_2 are respectively the masses, spring constants and the displacements, can be represented by a set of coupled second-order differential equations

$$\frac{\mathrm{d}^2 y_1}{\mathrm{d}t^2} = -5y_1 + 2y_2,$$

$$\frac{\mathrm{d}^2 y_2}{\mathrm{d}t^2} = 2y_1 - 2y_2.$$

Using vector notations so that $\mathbf{y} = (y_1, y_2)^{\mathrm{T}}$ and letting $\mathbf{y} = \mathbf{x}\,\mathrm{e}^{\mathrm{j}\omega t}$, where $\mathbf{x} = (x_1, x_2)^{\mathrm{T}}$ and $\mathrm{j} = \sqrt{-1}$, we obtain the matrix equation

$$-\omega^2 \mathbf{x} = \begin{pmatrix} -5 & 2 \\ 2 & -2 \end{pmatrix} \mathbf{x},$$

which can be reduced to the form

$$\mathbf{A}\mathbf{x} = \lambda \mathbf{x}. \tag{A.6}$$

The solution of (A.6) gives in turn the general solution for the original spring–mass system.

The importance of this example cannot be over-emphasized. The solutions of the homogeneous linear simultaneous equation (A.6) form a vector space in which all possible solutions of the original spring–mass system reside. (Think of these solutions as being generated by applying different initial disturbances to the point masses m_1 and m_2.) A direct application of ideas in the previous section means that if we can find a basis for this vector space, then all possible solutions are simply linear combinations of these basis vectors. To find these basis vectors, we shall continue with Example 17 and look for the solution space of (A.6).

Example 17: Solve the linear homogeneous system

$$\begin{pmatrix} -5 & 2 \\ 2 & -2 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \lambda \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}.$$

Rewriting these we obtain

$$\begin{pmatrix} -5-\lambda & 2 \\ 2 & -2-\lambda \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}, \tag{A.7}$$

where nontrivial solutions will exist only if the determinant of the coefficient matrix vanishes. Thus,

$$\det \begin{pmatrix} -5-\lambda & 2 \\ 2 & -2-\lambda \end{pmatrix} = (5+\lambda)(2+\lambda) - 4 = 0.$$

This is equivalent to the equation: $(\lambda+6)(\lambda+1)=0$. Therefore, the nontrivial solutions exist for $\lambda_1=-1$ and $\lambda_2=-6$. These are called *eigenvalues*. For $\lambda_1=-1$, (A.7) reduces to

$$\begin{pmatrix}-4 & 2\\ 2 & -1\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}0\\ 0\end{pmatrix}$$

or

$$\begin{pmatrix}x_1\\ x_2\end{pmatrix}=a\begin{pmatrix}1\\ 2\end{pmatrix}=a\mathbf{x}_1.$$

Note that the solution $a\mathbf{x}_1$ is determined up to an arbitrary constant a and $\mathbf{x}_1$ is referred to as the *eigenvector* corresponding to the eigenvalue $\lambda_1=-1$. For $\lambda_2=-6$, similar considerations will give

$$\begin{pmatrix}x_1\\ x_2\end{pmatrix}=b\begin{pmatrix}-2\\ 1\end{pmatrix}=b\mathbf{x}_2.$$

It is easy to show that the eigenvectors $\mathbf{x}_1$ and $\mathbf{x}_2$ are linearly independent. In fact, in this case, $<\mathbf{x}_1, \mathbf{x}_2>=0$, so that they are orthogonal to each other.

The various aspects of such a problem are dealt with in the following definition.

Definition 5: *Matrix eigenvalue problem*: Let $\mathbf{A}$ be an $n\times n$ matrix, $\mathbf{x}$ be an $n\times 1$ column vector and λ, a scalar, defined over a field F. Then the equation

$$\mathbf{A}\mathbf{x}=\lambda\mathbf{x} \tag{A.8}$$

is called a matrix eigenvalue problem for the matrix $\mathbf{A}$. The values of λ, for which nontrivial solutions exist are called eigenvalues. The corresponding solutions are called eigenvectors. All solutions of (A.8) (including the trivial solution $\mathbf{x}=\mathbf{0}$) form a vector space called the *eigen-space*. The set of eigenvalues λ's is called the *spectrum* of $\mathbf{A}$ and the largest absolute value of λ is called the *spectral radius* of A.

There are innumerable applications of eigenvalue problems in different engineering disciplines, and such applications necessitate a large amount of numerical methods for computers. Meanwhile, some of the more important analytical properties of the eigenvalue problem are considered.

Theorem 5: The eigenvalues of an $n\times n$ matrix are the roots of the so-called characteristic equation given by

$$D(\lambda)=\det|\mathbf{A}-\lambda\mathbf{I}|=0. \tag{A.9}$$

Equation (A.9) is simply the necessary and sufficient condition for the $n\times n$ homogeneous system to have nontrivial solutions. $D(\lambda)$, the characteristic polynomial, is one of n-th degree. The eigenvalues which are the roots of (A.9) can be real, complex, distinct or repeated as the roots of any polynomial equation can be. They are, however, often subject to physical interpretations. In Example 17, the eigenvalues are the squares of the angular frequencies of vibration of the system. The frequencies correspond to the two modes of

vibration exemplified by their corresponding eigenvectors. (One in which the masses move in phase and the other in which the masses move out of phase.) A typical 3×3 example is shown here for the evaluation of the eigenvalues.

Example 18: Find the eigenvalues of the 3×3 matrix $\mathbf{A}$ given by

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 2 \\ 1 & 1 & -1 \\ 2 & 3 & 0 \end{pmatrix}.$$

The characteristic polynomial is given by

$$\begin{aligned} D(\lambda) &= \det \begin{pmatrix} 2-\lambda & -1 & 2 \\ 1 & 1-\lambda & -1 \\ 2 & 3 & -\lambda \end{pmatrix}, \\ &= (2-\lambda)\{-(1-\lambda)\lambda + 3\} + \{-\lambda + 2\} + 2\{3 - 2(1-\lambda)\} \\ &= \lambda^3 - 3\lambda^2 + 10 = 0. \end{aligned}$$

The roots of this cubic are 3.3089, $-0.1545 + \mathrm{j}1.7316$ and $-0.1545 - \mathrm{j}1.7316$.

This example shows that even when the matrix $\mathbf{A}$ is defined over the field of real numbers $\boldsymbol{R}$, the eigenvalues, and therefore the eigenvectors may be defined over the field of complex numbers $\boldsymbol{C}$. The subject of matrix eigenvalue problem is a very rich one and there are many interesting properties associated with the eigenvalues of a given matrix $\mathbf{A}$. Some important ones are listed here.

Properties of eigenvalues: Let $\lambda_1, \lambda_2, \ldots, \lambda_n$ be the eigenvalues of an $n \times n$ matrix $\mathbf{A}$. Then the following properties hold:

1. *Trace*: $\mathrm{Tr}(\mathbf{A}) = \sum_{i=1}^{n} a_{ii} = \sum_{i=1}^{n} \lambda_i$.
2. *Determinant*: $\det |\mathbf{A}| = \prod_{i=1}^{n} \lambda_i$.
3. *Inverse*: If $\mathbf{A}$ is nonsingular, then the eigenvalues of $\mathbf{A}^{-1}$ are $\lambda_1^{-1}, \lambda_2^{-1}, \ldots, \lambda_n^{-1}$.
4. *Spectral shift*: The matrix $(\mathbf{A} - k\mathbf{I})$ has eigenvalues $\lambda_1 - k, \ldots, \lambda_n - k$, for a constant k.
5. *Spectral mapping*: The matrix $k_m\mathbf{A}^m + k_{m-1}\mathbf{A}^{m-1} + \cdots + k_1\mathbf{A} + k_0\mathbf{I}$ has eigenvalues
$$k_m\lambda_j^m + k_{m-1}\lambda_j^{m-1} + \cdots + k_1\lambda_j + k_0, \quad j = 1, 2, \ldots, n.$$

More important are the eigenvalue problems of matrices of special symmetries. The definitions for some of these special matrices are given here.

Definition 6: *Hermitian matrix*: A square matrix **A** defined over the field of complex numbers ***C*** is said to be Hermitian if

$$a_{ij} = a_{ji}^*$$

and is said to be skew-Hermitian if

$$a_{ij} = -a_{ji}^*,$$

where the asterisk * denotes complex conjugation. In matrix notations **A** is Hermitian if

$$\mathbf{A}^{\mathrm{T}} = \mathbf{A}^*,$$

and is skew-Hermitian if

$$\mathbf{A}^{\mathrm{T}} = -\mathbf{A}^*.$$

In the special case where the matrix elements are all real, these two reduce to being symmetric and skew-symmetric, respectively.

Definition 7: *Unitary matrix*: A nonsingular square matrix **A** is unitary if

$$\mathbf{A}^{-1} = (\mathbf{A}^*)^{\mathrm{T}} (= \mathbf{A}^{\mathrm{H}}),$$

where the superscript H denotes conjugate transposition.

Here are examples of the special matrices.

Example 19:

$$\mathbf{A} = \begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & 2 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & \mathrm{j} \\ \mathrm{j} & 0 \end{pmatrix}, \quad \mathbf{C} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & -1 \end{pmatrix}.$$

Note that $\mathbf{A}^{\mathrm{T}} = \mathbf{A}^*$, therefore **A** is Hermitian (you note that a_{ii}'s are real). $\mathbf{B}^{\mathrm{T}} = -\mathbf{B}^*$, therefore **B** is skew-Hermitian (you note that b_{ii}'s are zero). $\mathbf{C}^{\mathrm{H}}\mathbf{C} = \mathbf{I}$, therefore **C** is unitary.

The following is a very important result about the eigenvalues of these special matrices.

Theorem 6: *Eigenvalues of special matrices*:

1. The eigenvalues of a Hermitian matrix are real.
2. The eigenvalues of a skew-Hermitian matrix are purely imaginary or zero.
3. The eigenvalues of a unitary matrix have absolute value 1.

Proof: The proof of (1) above is instructive and is outlined here.

Let $\mathbf{A}$ be a square Hermitian matrix so that $\mathbf{A}^{\mathrm{T}} = \mathbf{A}^*$, and let the eigenvalue be λ, so that

$$\mathbf{A}\mathbf{x} = \lambda\mathbf{x}.$$

Premultiply this equation by $\mathbf{x}^{\mathrm{H}}$ to get

$$\mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x} = \lambda\mathbf{x}^{\mathrm{H}}\mathbf{x}, \quad \text{so that } \lambda = (\mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x})/\mathbf{x}^{\mathrm{H}}\mathbf{x}.$$

Now, $\mathbf{x}^{\mathrm{H}}\mathbf{x}$, the magnitude squared of the eigenvector $\mathbf{x}$ is of course real. The quantity $\mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x}$ is also real since,

$$(\mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x})^{\mathrm{H}} = \mathbf{x}^{\mathrm{H}}\mathbf{A}^{\mathrm{H}}(\mathbf{x}^{\mathrm{H}})^{\mathrm{H}} = \mathbf{x}^{\mathrm{H}}\mathbf{A}\mathbf{x}.$$

Thus λ must be real. The proofs of (2) and (3) in Theorem 6 are left as exercises. The theorem is valid also when the matrix elements are real, giving rise to symmetric and skew-symmetric matrices for (1) and (2).

Eigenvectors also form a very rich subject. To this subject we now turn our attention first by introducing the concept of *similarity* between matrices.

Definition 8: *Similarity of matrices*: An $n \times n$ matrix $\mathbf{A}$ is said to be similar to a matrix $\mathbf{B}$ if they are related by the equation

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}, \tag{A.10}$$

where $\mathbf{T}$ is some nonsingular $n \times n$ transformation matrix. Matrix $\mathbf{A}$ is said to have undergone a similarity transformation in (A.10).

Example 20: The matrix $\mathbf{A} = \begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & 2 \end{pmatrix}$ is similar to the matrix

$$\mathbf{B} = \frac{1}{2}\begin{pmatrix} 5 & -\mathrm{j} \\ \mathrm{j} & 1 \end{pmatrix},$$

since

$$\mathbf{B} = \frac{1}{\sqrt{2}}\begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & -1 \end{pmatrix}\begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & 2 \end{pmatrix}\frac{1}{\sqrt{2}}\begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & -1 \end{pmatrix} = \mathbf{C}^{-1}\mathbf{A}\mathbf{C}.$$

Note in this example that the characteristic equation for the matrix $\mathbf{A}$ is given by

$$\det|\mathbf{A} - \lambda\mathbf{I}| = (1-\lambda)(2-\lambda) - 1 = \lambda^2 - 3\lambda + 1 = 0.$$

The characteristic equation for the matrix $\mathbf{B}$ is given by

$$\det|\mathbf{B} - \lambda\mathbf{I}| = (2.5-\lambda)(0.5-\lambda) - 0.25 = \lambda^2 - 3\lambda + 1 = 0.$$

Thus, it is seen that **A** and **B** have exactly the same eigenvalues. This is a special case of a more general result stated in the following theorem.

Theorem 7: If $\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$, then **B** has the same eigenvalues as **A**. Also, if **x** is an eigenvector of **A**, then $\mathbf{T}^{-1}\mathbf{x}$ is an eigenvector of **B**.

Proof: The proof is straightforward and will be outlined here.

Let $\mathrm{A}\mathbf{x} = \lambda\mathbf{x}$ be the eigenvalue problem for the matrix **A**, and let **B** be similar to **A** by the transformation $\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T}$. We can apply the transformation $\mathbf{T}^{-1}$ to the equation to obtain

$$\mathbf{T}^{-1}\mathbf{A}(\mathbf{T}\mathbf{T}^{-1})\mathbf{x} = \lambda\mathbf{T}^{-1}\mathbf{x},$$

which is the same as:

$$(\mathbf{T}^{-1}\mathbf{A}\mathbf{T})(\mathbf{T}^{-1}\mathbf{x}) = \mathbf{B}(\mathbf{T}^{-1}\mathbf{x}) = \lambda(\mathbf{T}^{-1}\mathbf{x}) \tag{A.11}$$

which clearly states that the vector $(\mathbf{T}^{-1}\mathbf{x})$ is an eigenvector of the matrix **B**.

Eigenvalues are very often physical quantities in a system being analyzed, and, therefore, similar matrices represent in many ways equivalent physical systems. In particular, when the eigenvectors of a matrix **A** are linearly independent, they can be used to form a transformation **T**. The matrix **A** is then similar to the diagonal matrix with the eigenvalues along the diagonal, through this transformation matrix **T**. This, we show in our next example.

Example 21: Recall the matrix in Example 17 where

$$\mathbf{A} = \begin{pmatrix} -5 & 2 \\ 2 & -2 \end{pmatrix} \quad \text{and} \quad \lambda_1 = -1, \lambda_2 = -6.$$

The eigenvectors are found to be $\mathbf{x}_1 = (1, 2)^{\mathrm{T}}$ and $\mathbf{x}_2 = (-2, 1)^{\mathrm{T}}$. Let

$$\mathbf{T} = (\mathbf{x}_1, \mathbf{x}_2) = \begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}.$$

Then,

$$\mathbf{T}^{-1} = \frac{1}{5}\begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}.$$

We find that

$$\mathbf{B} = \mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \frac{1}{5}\begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} -5 & 2 \\ 2 & -2 \end{pmatrix}\begin{pmatrix} 1 & -2 \\ 2 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -6 \end{pmatrix}.$$

Such a relation as shown in Example 21 only exists if the eigenvectors are linearly independent, forming a basis for the eigen-space for **A**. This situation is assured when the eigenvalues are all distinct. The result is stated more formally in the following theorem.

Theorem 8: If the eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$ of a matrix **A** are distinct, the corresponding eigenvectors $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n$ are linearly independent and form a basis for the eigen-space of **A**.

In fact, basis of eigenvectors exists under much weaker condition than is stated in Theorem 8. In other words, it is possible under certain conditions for a linearly independent set of eigenvectors to exist even when the eigenvalues are not all distinct. This is illustrated in our next example.

Example 22: Consider the 3×3 matrix **A** given by

$$\mathbf{A} = \begin{pmatrix} 2 & 0 & 2 \\ -1 & 1 & -2 \\ 0 & 0 & 1 \end{pmatrix} \text{ whose eigenvalues are } \lambda_1 = \lambda_2 = 1 \text{ and } \lambda_3 = 2,$$

as can easily be verified. To find the eigenvector for $\lambda_3 = 2$, we have $\mathbf{A}\mathbf{x}_3 = 2\mathbf{x}_3$, or

$$\begin{pmatrix} 0 & 0 & 2 \\ -1 & -1 & -2 \\ 0 & 0 & -1 \end{pmatrix} \mathbf{x}_3 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

giving $\mathbf{x}_3 = (1, -1, 0)^{\mathrm{T}}$. For $\lambda_1 = \lambda_2 = 1$,

$$\begin{pmatrix} 1 & 0 & 2 \\ -1 & 0 & -2 \\ 0 & 0 & 0 \end{pmatrix} \mathbf{x}_i = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix},$$

giving $\mathbf{x}_i = a(-2, b, 1)^{\mathrm{T}}$, $i = 1, 2$.

By choosing $a = 1$, $b = 2$ for $\mathbf{x}_1$ and $a = -1$, $b = 1$ for $\mathbf{x}_2$, $\mathbf{x}_1 = (-2,\ 2,\ 1)^{\mathrm{T}}$, $\mathbf{x}_2 = (2,\ -1,\ -1)^{\mathrm{T}}$ and together with $\mathbf{x}_3$ above, they form a linearly independent set. Using these eigenvectors we can form the transformation matrix **T** given by

$$\mathbf{T} = \begin{pmatrix} -2 & 2 & 1 \\ 2 & -1 & -1 \\ 1 & -1 & 0 \end{pmatrix} \quad \text{and} \quad \mathbf{T}^{-1} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 2 \end{pmatrix}.$$

From these, it is not difficult to verify that

$$\mathbf{T}^{-1}\mathbf{A}\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 2 \end{pmatrix}.$$

Thus the existence of **T** to "diagonalize" **A** depends on whether it is possible to find linearly independent eigenvectors for the repeated eigenvalues. It turns out that for matrices of certain special symmetries it is always possible. The following theorem illustrates some of these cases.

Theorem 9: An $n \times n$ Hermitian, skew-Hermitian or unitary matrix has a set of linearly independent eigenvectors (defined over $\boldsymbol{R}^n$ or $\boldsymbol{C}^n$), which form a unitary transformation matrix **T**. In particular, a symmetric matrix has a set of orthonormal eigenvectors. Symbolically, we have

$$\mathbf{T}^{\mathrm{H}}\mathbf{A}\mathbf{T} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) \quad \text{when } \mathbf{A} \text{ is Hermitian, skew-Hermitian or unitary;}$$

$$\text{and} \quad \mathbf{T}^{\mathrm{T}}\mathbf{A}\mathbf{T} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n), \quad \text{when } \mathbf{A} \text{ is symmetric.} \tag{A.12}$$

Here, we have used the convention of $\mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ to denote a diagonal matrix made up of eigenvalues $\lambda_1, \ldots, \lambda_n$.

One of the most important and in our case most relevant diagonalization problems is that of a circulant matrix. The matrix which diagonalizes a circulant matrix is exactly the matrix representing the discrete Fourier transform (DFT). We will first review the definition of a circulant matrix and then discuss its diagonalization as a concluding example of this section.

Definition 9: *Circulant matrix*: A circulant matrix **C** of order n is an $n \times n$ matrix in which the elements of each row are identical to those of the previous row, but are moved one position to the right and wrapped around. Thus,

$$\mathbf{C} = \begin{pmatrix} c_1 & c_2 & \ldots & c_n \\ c_n & c_1 & \ldots & c_{n-1} \\ \vdots & \vdots & \vdots & \vdots \\ c_2 & c_3 & \ldots & c_1 \end{pmatrix} \text{ is a circulant matrix.} \tag{A.13}$$

Note that **C** is also Toeplitz, i.e., each main diagonal consists of the same element.

Example 23: In this example, the circulant matrix of order n and its diagonalization are examined. Let **C** be a circulant matrix of order n. Define a matrix $\mathbf{F}^*$ whose elements are given by

$$(\mathbf{F}^*)_{km} = \frac{1}{\sqrt{n}} W_n^{(k-1)(m-1)}, \quad k, m = 1, \ldots, n$$

and $W_n = \exp(-2\pi \mathrm{j}/n) = \cos(2\pi/n) - \mathrm{j}\,\sin(2\pi/n)$ is the primitive n-th root of unity. Note that $\mathbf{F}^*$ is unitary and is the inverse of the n-th-order DFT. Let us look at the k-th column

of the matrix. This is given by the column vector $\mathbf{f}_k$:

$$\mathbf{f}_k = \frac{1}{\sqrt{n}}(1, W_n^{(k-1)}, W_n^{2(k-1)}, \ldots, W_n^{(n-1)(k-1)})^{\mathrm{T}}.$$

Consider now the product $\mathbf{C}\mathbf{f}_k$, denoted by the vector $\mathbf{g}_k$. The first element of this vector, denoted by the scalar g_{1k}, is

$$\begin{aligned} g_{1k} &= \frac{1}{\sqrt{n}}(c_1 + c_2 W_n^{(k-1)} + c_3 W_n^{2(k-1)} + \cdots + c_n W_n^{(n-1)(k-1)}) \\ &= \frac{1}{\sqrt{n}}(c_1 + c_2 z + c_3 z^2 + \cdots + c_n z^{n-1}) = p_k(z). \end{aligned}$$

Note that we have defined $W_n^{(k-1)}$ as z and this first element is simply an $(n-1)$-th degree polynomial of z as defined by $p_k(z)$.

The second element of the vector $\mathbf{g}_k$, denoted by g_{2k}, is

$$\begin{aligned} g_{2k} &= \frac{1}{\sqrt{n}}(c_n + c_1 z + \cdots + c_{n-1} z^{n-1}) \\ &= \frac{1}{\sqrt{n}} z(c_1 + c_2 z + c_3 z^2 + \cdots + c_n z^{n-1}) \text{ since } z^n = 1, \\ &= z p_k(z). \end{aligned}$$

Similarly, the m-th element g_{mk} of the vector $\mathbf{g}_k$ is given by

$$g_{mk} = z^{m-1} p_k(z).$$

Therefore, the vector $\mathbf{g}_k$ is given by

$$\begin{aligned} \mathbf{g}_k &= p_k(W_n^{(k-1)})(1, W_n^{(k-1)}, W_n^{2(k-1)}, \ldots, W_n^{(n-1)(k-1)})^{\mathrm{T}} \\ &= p_k(W_n^{(k-1)})\mathbf{f}_k. \end{aligned}$$

Thus, $\mathbf{C}\mathbf{f}_k = p_k(W_n^{(k-1)})\mathbf{f}_k$, which is saying that the vector $\mathbf{f}_k$ is an eigenvector for the circulant matrix $\mathbf{C}$. Since the vectors $\mathbf{f}_k$, $k = 1, \ldots, n$ make up the unitary matrix $\mathbf{F}^*$, it is clear that the matrix $\mathbf{F}^*$ will diagonalize the circulant $\mathbf{C}$, or

$$(\mathbf{F}^*)^{-1}\mathbf{C}(\mathbf{F}^*) = \mathrm{diag}(p_1(W_n^0),\ p_2(W_n^1),\ p_3(W_n^2), \ldots, p_n(W_n^{n-1})). \tag{A.14}$$

Note that the scalar polynomials p_i need not be distinct, and while the actual elements in $\mathbf{C}$ will affect the eigenvalues p_i, they have no effect on the eigenvectors, so long as $\mathbf{C}$ is circulant. Note also that the circulant matrix, which is not one of the special types included in Theorem 8, is always diagonalizable too. Therefore, Theorem 9 is not exhaustive.

Problems and Exercises A.2

1. Find the eigenvalues for the following matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & 2 \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} 0 & \mathrm{j} \\ \mathrm{j} & 0 \end{pmatrix}, \quad \mathbf{C} = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & \mathrm{j} \\ -\mathrm{j} & -1 \end{pmatrix},$$

   and verify the statements in Theorem 6.

2. Prove, for an $n \times n$ matrix $\mathbf{A}$, where n is odd, that there is at least one real eigenvalue, if all elements of $\mathbf{A}$ are real.

3. Prove, for an $n \times n$ matrix $\mathbf{A}$ defined over the field of real numbers $\boldsymbol{R}$, that the complex eigenvalues will appear in complex conjugate pairs.

4. Consider the coupled differential equations

$$\frac{\mathrm{d}^2 y_1}{\mathrm{d}t^2} = -5y_1 + 2y_2 + 1,$$
$$\frac{\mathrm{d}^2 y_2}{\mathrm{d}t^2} = 2y_1 - 2y_2 + 2.$$

   which can be written in matrix form $\dfrac{\mathrm{d}^2 \mathbf{y}}{\mathrm{d}t^2} = \mathbf{A}\mathbf{y} + \mathbf{b}$.

   Find a time-independent transformation matrix $\mathbf{T}$, so that $\mathbf{z} = \mathbf{T}^{-1}\mathbf{y}$ and $\mathbf{T}^{-1}\,\mathbf{A}\mathbf{T} = \mathbf{D}$ is diagonal.

   Apply the matrix $\mathbf{T}^{-1}$ to get the de-coupled system

$$\frac{\mathrm{d}^2 \mathbf{z}}{\mathrm{d}t^2} = \mathbf{D}\mathbf{z} + \mathbf{T}^{-1}\mathbf{b}.$$

   Solve this de-coupled system and find the solution to the original equations.

5. Prove, for an $n \times n$ matrix $\mathbf{A}$ with eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$, that

$$\text{(a) Trace } (\mathbf{A}) = \sum_{i=1}^{n} \lambda_i \quad \text{and} \quad \text{(b) } \det|\mathbf{A}| = \prod_{i=1}^{n} \lambda_i.$$

   In particular, show that at least one of the eigenvalues for a singular matrix must be zero.

6. Prove, if $\mathbf{A}$ is an $n \times n$ triangular (upper or lower) matrix, that its eigenvalues are its diagonal elements, i.e.,

$$\lambda_i = a_{ii}, \quad i = 1, 2, \ldots, n.$$

7. If the elements of the first row in an $n \times n$ circulant matrix $\mathbf{C}$ are

$$c_1, c_2, c_3, \ldots, c_n,$$

   what conditions must be satisfied by these elements so that

(a) **C** is symmetric,

(b) **C** is Hermitian,

(c) **C** is skew-Hermitian.

8. Prove that if k is a positive integer and the matrix **A** is circulant, $\mathbf{A}^k$ is also circulant. In addition, if **A** is also nonsingular, then $\mathbf{A}^{-k}$ is also circulant.

9. Diagonalize the $n \times n$ circulant matrix **J**, whose first row is $(1, 1, 1, \ldots, 1)$.

10. In Example 23 it is shown that the k-th eigenvalue of an $n \times n$ circulant matrix **C** is given by

$$\mathrm{p}_k(z) = (c_1 + c_2 z + c_3 z^2 + \cdots + c_n z^{n-1}),$$

where $z = \exp[\mathrm{j}2\pi(k-1)/n]$ and c_i is the i-th element of **C** in the first row. Use this result to find the determinant of the circulant matrix whose first row is $(x, 1, 1, 1)$, where x is any real number.

A.3 Matrix decompositions

Matrices occur most naturally in problems involving simultaneous equations. We have seen that a set of second-order differential equations describing the motion of a coupled spring–mass system can be reduced to matrix form. Its solution is intimately related to the eigenvalue decomposition problem that we describe in the last section. Looking at the eigenvalue problem as one of diagonalization, we can see why the equation

$$\mathbf{A} = \mathbf{T}\mathbf{D}\mathbf{T}^{-1}, \tag{A.15}$$

where **D** is a diagonal matrix, can be considered as one of decomposing the matrix **A** into factor matrices. Of course, it may not be always possible to achieve the decomposition in this form. In fact, the relevant form of decomposition depends critically on the purpose of the decomposition. In this section we shall examine some other forms of decomposition for matrices.

As is our custom, an example is used to begin the discussion.

Example 24: Consider the equation

$$\mathbf{A}\mathbf{x} = \mathbf{b}, \tag{A.16}$$

where

$$\mathbf{A} = \begin{pmatrix} 0 & 8 & 2 \\ 3 & 5 & 2 \\ 6 & 2 & 8 \end{pmatrix},$$

and $\mathbf{b} = (-7, 8, 26)^{\mathrm{T}}$, which has the solution $\mathbf{x} = (4, -1, 0.5)^{\mathrm{T}}$, as can be easily verified. The actual solution of (A.16) is obtained by a combination of Gauss eliminations and back

substitutions. The idea is neatly summarized in the following schematic diagram:

$$\begin{aligned}&(\mathbf{A},\mathbf{b}) \rightarrow \text{by Gauss eliminations} \rightarrow (\mathbf{U},\mathbf{b}')\\ &(\mathbf{U},\mathbf{b}') \rightarrow \text{by back substitutions} \rightarrow (\mathbf{I},\mathbf{x}),\end{aligned} \tag{A.17}$$

where $\mathbf{U}$ is an "upper triangular" matrix. In our case, after the proper Gauss eliminations we have for (A.16) the upper triangular system

$$\begin{pmatrix}3 & 5 & 2\\ 0 & 8 & 2\\ 0 & 0 & 6\end{pmatrix}\begin{pmatrix}x_1\\ x_2\\ x_3\end{pmatrix} = \begin{pmatrix}8\\ -7\\ 3\end{pmatrix},$$

from which we get $x_3 = 0.5, x_2 = -1$ and $x_3 = 4$ by back substitutions.

In this example, the Gauss elimination steps can be summarized by a matrix $\mathbf{G}$ with the result that

$$\mathbf{GA} = \mathbf{U}. \tag{A.18}$$

If $\mathbf{G}$ is nonsingular, we have the equivalent relation

$$\mathbf{A} = \mathbf{G}^{-1}\mathbf{U} = \mathbf{L}'\mathbf{U}, \tag{A.19}$$

which is saying that $\mathbf{A}$ has been factorized into two factor matrices, one of which is upper triangular. In the case of Example 24, we have

$$\mathbf{A} = \begin{pmatrix}0 & 1 & 0\\ 1 & 0 & 0\\ 2 & -1 & 1\end{pmatrix}\begin{pmatrix}3 & 5 & 2\\ 0 & 8 & 2\\ 0 & 0 & 6\end{pmatrix}.$$

Suppose the first two rows in $\mathbf{A}$ are interchanged. This is equivalent to interchanging the first two equations in (A.16), resulting in an exactly equivalent system of equations. Then,

$$\mathbf{PA} = \begin{pmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 2 & -1 & 1\end{pmatrix}\begin{pmatrix}3 & 5 & 2\\ 0 & 8 & 2\\ 0 & 0 & 6\end{pmatrix} = \mathbf{LU}, \tag{A.20}$$

where $\mathbf{P}$ is a 3×3 permutation matrix representing the interchange of the first two rows. Note that (A.20) indicates that $\mathbf{PA}$ can be factorized into a lower triangular matrix $\mathbf{L}$ and an upper triangular matrix $\mathbf{U}$. This is in fact true for any nonsingular matrix. The following theorem on $\mathbf{LU}$ decomposition states this result more formally.

Theorem 10: For any nonsingular matrix $\mathbf{A}$, the rows of $\mathbf{A}$ can be permuted so that the resulting matrix $\mathbf{PA}$ can be factorized into the form $\mathbf{PA} = \mathbf{LU}$, where $\mathbf{L}$ and $\mathbf{U}$ are respectively lower triangular and upper triangular. In particular, if the diagonal elements of $\mathbf{L}$ are ones, the factors $\mathbf{L}$ and $\mathbf{U}$ are unique (the *Doolittle factorization*). If instead, the

diagonal elements of **U** are ones, the factorization is called *Crout's factorization* which is also unique.

Instead of proving the theorem, it is more illuminating to examine the consequences of the theorem. Let

$$\mathbf{PAx} = \mathbf{Pb}, \tag{A.21a}$$

be a set of equations in which the required row interchange operations have taken place. Using the theorem it is seen that

$$\mathbf{L}(\mathbf{Ux}) = \mathbf{Pb}, \tag{A.21b}$$

which can be separated into two sets of simultaneous equations

$$\mathbf{Ly} = \mathbf{Pb} \quad \text{and} \quad \mathbf{Ux} = \mathbf{y}. \tag{A.22}$$

It is apparent that **y** can be found from the first equation in (A.22) by forward substitution, since **L** is lower triangular. Using the solution **y** in the second equation, **x** can be obtained by backward substitution, since **U** is upper triangular. Generally, the forward and backward substitutions in (A.22) are more easily accomplished computationally than the direct inversion of the matrix **PA**.

While all nonsingular matrices have LU decompositions, given the required row operations, symmetric positive definite matrices have LU decompositions which have additional symmetries. Suppose **A** is symmetric positive definite[1] and has an LU decomposition given by **A** = **LU**, then it is easy to see that

$$\mathbf{A}^{\mathrm{T}} = \mathbf{U}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}} = \mathbf{A} = \mathbf{LU}, \tag{A.23}$$

from which we can choose a decomposition for **A** such that

$$\mathbf{L} = \mathbf{U}^{\mathrm{T}}. \tag{A.24}$$

In this particular case, the requirement that either **U** or **L** should have unit diagonal elements must be removed. The resulting decomposition is called the *Cholesky's decomposition*. An example of the process of this decomposition follows.

Example 25: Consider the symmetric positive definite matrix **A**

$$\mathbf{A} = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 10 & 5 \\ 4 & 5 & 21 \end{pmatrix}.$$

The LU decomposition in this case is in the following form

$$\mathbf{A} = \mathbf{LL}^{\mathrm{T}} = \begin{pmatrix} l_{11} & 0 & 0 \\ l_{21} & l_{22} & 0 \\ l_{31} & l_{32} & l_{33} \end{pmatrix} \begin{pmatrix} l_{11} & l_{21} & l_{31} \\ 0 & l_{22} & l_{32} \\ 0 & 0 & l_{33} \end{pmatrix},$$

[1] **A** is positive definite matrix if $\mathbf{x}^{\mathrm{T}}\mathbf{Ax} > 0$ for all $\mathbf{x} \neq \mathbf{0}$.

from which the following equations can be obtained for the elements of the matrix **L**

$$l_{11}l_{11} = 4, \quad l_{11} = 2. \qquad l_{11}l_{21} = 2, \quad l_{21} = 1.$$
$$l_{11}l_{31} = 4, \quad l_{31} = 2. \qquad l_{21}^2 + l_{22}^2 = 10, \quad l_{22} = 3.$$
$$l_{31}l_{21} + l_{32}l_{22} = 5, \quad l_{32}l_{22} = 3, \quad l_{32} = 1.$$
$$l_{31}^2 + l_{32}^2 + l_{33}^2 = 21, \quad l_{33} = 4.$$

Hence,

$$\mathbf{L} = \begin{pmatrix} 2 & 0 & 0 \\ 1 & 3 & 0 \\ 2 & 1 & 4 \end{pmatrix}.$$

It can be seen in this example that the matrix **L** obtained as a result of this procedure is not unique because of the square root operations. However, if the diagonal elements of **L** are chosen to be positive this Cholesky decomposition is unique. When the matrix **A** is just symmetric (but not positive definite), **L** will be a matrix with complex elements. In relation to solving systems of linear equations, it is seen that when **A** is symmetric and positive definite the Cholesky decomposition does provide a very compact and computationally efficient way for solving the linear equations.

When a matrix **A** is similar to a diagonal matrix **D**, and at the same time can be LU decomposed so that the diagonal elements of **L** are ones, its eigenvalues and the diagonal elements of **U** are very intimately related. Because **A** is similar to **D** and $\det|\mathbf{L}| = 1$, the following equalities hold:

$$\det|\mathbf{A}| = \det|\mathbf{U}| = \det|\mathbf{D}|. \tag{A.25}$$

Alternatively, denoting the eigenvalues of **A** by λ_i,

$$\prod_i u_{ii} = \prod_i d_{ii} = \prod_i \lambda_i. \tag{A.26}$$

The question which leads to interesting results is whether or not the λ_i's can be identified with the individual u_{ii}'s. If so, the eigenvalue problem for **A** can be efficiently solved by some type of LU decomposition. The answer, however, is not as straightforward as one would like it to be. To explore this connection, a different orthogonal decomposition of the matrix **A** needs to be considered.

In this decomposition, a real matrix **A** is factorized into two factor matrices **Q** and **R**, so that $\mathbf{A} = \mathbf{QR}$. **Q** is an orthogonal normalized (unitary) matrix so that $\mathbf{Q}^{\mathrm{T}}\mathbf{Q} = \mathbf{I}$, and **R** is an upper triangular matrix. Such a decomposition is called a QR decomposition and the procedure to obtain the factors **Q** and **R** can be illustrated by what follows. The idea is to apply orthogonal operations to **A** so that the final result will be an upper triangular matrix **R**. One can systematically do this by working on one column at a time in the matrix **A.**

For the first column in the matrix

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{pmatrix},$$

we apply an orthogonal operation defined as a premultiplication by a matrix $\mathbf{C}_1$ so that the element a_{21} will become zero. This can be accomplished by using what is called a *Givens rotation*. Such a matrix is given by

$$\mathbf{C}_1 = \begin{pmatrix} c_1 & -s_1 & \dots & \dots \\ s_1 & c_1 & 1 & \dots \\ \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & 1 \end{pmatrix},$$

where the elements are all zeros except along the diagonal where they are ones, and the first four elements in the upper left are as shown. We require that the product $\mathbf{C}_1\mathbf{A}$ to have a zero for the first element in the second row. This and the requirement that the matrix $\mathbf{C}_1$ be unitary give the following equations

$$a_{11}s_1 + a_{21}c_1 = 0 \quad \text{and} \quad c_1^2 + s_1^2 = 1.$$

It can be seen easily that if $c_1 = \cos\theta$ and $s_1 = \sin\theta$, the matrix $\mathbf{C}_1$ is a rotation matrix, and the angle of rotation can be determined from the two equations. Similarly the element a_{31} can be eliminated by applying a second unitary matrix $\mathbf{C}_2$ given by

$$\mathbf{C}_2 = \begin{pmatrix} c_2 & \dots & -s_2 & \dots \\ \dots & 1 & \dots & \dots \\ s_2 & \dots & c_2 & \dots \\ \dots & \dots & \dots & \dots \end{pmatrix}.$$

After all the elements below a_{11} have been nulled, the procedure is repeated for the $(n-1)\times(n-1)$ submatrix below the first row, to eliminate all the elements below a_{22}. The process is continued until all elements below the main diagonal are eliminated so that the result is an upper triangular matrix $\mathbf{R}$. The result is stated as

$$(\cdots \mathbf{C}_{n-1}\mathbf{C}_{n-2}\cdots\mathbf{C}_2\mathbf{C}_1)\mathbf{A} = \mathbf{R}, \tag{A.27}$$

and since all the $\mathbf{C}$'s are unitary, (A.27) can be written as

$$\mathbf{A} = (\cdots \mathbf{C}_{n-1}\mathbf{C}_{n-2}\cdots\mathbf{C}_2\mathbf{C}_1)^{\mathrm{T}}\mathbf{R} = \mathbf{Q}\mathbf{R}. \tag{A.28}$$

It is seen that since each $\mathbf{C}$ is orthogonal, the transpose of the product of these matrices is also orthogonal, giving $\mathbf{Q}$ with the required property. The procedure for the QR decomposition of a symmetric real matrix is illustrated in the next example.

Example 26: Find the QR decomposition for the real symmetric matrix $\mathbf{A}$ given by

$$\mathbf{A} = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 10 & 5 \\ 4 & 5 & 21 \end{pmatrix}.$$

Step 1: Find $\mathbf{C}_1 = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix}$ such that the element $[\mathbf{C}_1\mathbf{A}]_{21} = 0$ and $c^2 + s^2 = 1$; the nulling to the *21*-element gives: $4s + 2c = 0$. These two equations for c and s provide one possible solution of $s = -1/\sqrt{5}$ and $c = 2/\sqrt{5}$. Note that c and s are not unique. Thus $\mathbf{A}_1$ the matrix after this operation is:

$$\mathbf{A}_1 = \mathbf{C}_1\mathbf{A} = \begin{pmatrix} c & -s & 0 \\ s & c & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} 4 & 2 & 4 \\ 2 & 10 & 5 \\ 4 & 5 & 21 \end{pmatrix} = \begin{pmatrix} \frac{10}{\sqrt{5}} & \frac{14}{\sqrt{5}} & \frac{13}{\sqrt{5}} \\ 0 & \frac{18}{\sqrt{5}} & \frac{6}{\sqrt{5}} \\ 4 & 5 & 21 \end{pmatrix}.$$

Step 2: Find $\mathbf{C}_2 = \begin{pmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{pmatrix}$ such that $[\mathbf{C}_2\mathbf{A}_1]_{31} = 0$. Hence, $(10/\sqrt{5})s + 4c = 0$, and $c^2 + s^2 = 1$. One possible solution is, $s = -2/3$ and $c = \sqrt{5}/3$. The result of this operation on $\mathbf{A}_1$ is

$$\mathbf{A}_2 = \mathbf{C}_2\mathbf{A}_1 = \begin{pmatrix} c & 0 & -s \\ 0 & 1 & 0 \\ s & 0 & c \end{pmatrix} \begin{pmatrix} \frac{10}{\sqrt{5}} & \frac{14}{\sqrt{5}} & \frac{13}{\sqrt{5}} \\ 0 & \frac{18}{\sqrt{5}} & \frac{6}{\sqrt{5}} \\ 4 & 5 & 21 \end{pmatrix} = \begin{pmatrix} 6 & 8 & \frac{55}{3} \\ 0 & \frac{18}{\sqrt{5}} & \frac{6}{\sqrt{5}} \\ 0 & \frac{-1}{\sqrt{5}} & \frac{79}{\sqrt{45}} \end{pmatrix}$$

Step 3: Find $\mathbf{C}_3 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix}$ such that $[\mathbf{C}_3\mathbf{A}_2]_{32} = 0$. In this case, the solution $s = -1/\sqrt{325}$ and $c = -18/\sqrt{325}$ is a possibility. The final matrix $\mathbf{R}$ which is upper triangular is given by

$$\mathbf{R} = \mathbf{C}_3\mathbf{A}_2 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & c & -s \\ 0 & s & c \end{pmatrix} \begin{pmatrix} 6 & 8 & \frac{55}{3} \\ 0 & \frac{18}{\sqrt{5}} & \frac{6}{\sqrt{5}} \\ 0 & \frac{-1}{\sqrt{5}} & \frac{79}{\sqrt{45}} \end{pmatrix} = \begin{pmatrix} 6 & 8 & \frac{55}{3} \\ 0 & -\sqrt{65} & \frac{-49}{\sqrt{195}} \\ 0 & 0 & \frac{-473}{\sqrt{325}} \end{pmatrix}.$$

The orthogonal matrix $\mathbf{Q}$ is just the transpose of the product $\mathbf{C}_3\mathbf{C}_2\mathbf{C}_1$ and is given to four decimal places by

$$\mathbf{Q} = \begin{pmatrix} 0.6667 & 0.4134 & 0.6202 \\ 0.3333 & -0.9096 & 0.2481 \\ 0.6667 & 0.0413 & -0.7442 \end{pmatrix}.$$

It should be noted that since the elements in the matrices $\mathbf{C}$'s are determined up to a sign, the QR decomposition cannot be considered as unique. As indicated earlier, the $\mathbf{C}$ matrices can be considered as rotation matrices so that the elements c and s are respectively cosine and sine of an angle of rotation θ. The question of whether such Givens rotations will always exist is examined in the following.

Consider the lm-th element of a matrix $\mathbf{A}$ to be nulled in its QR decomposition. (note that $l > m$ since the element is below the main diagonal). It is only necessary to consider the 2×2 matrix made up of a_{ll}, a_{lm}, a_{ml} and a_{mm}, i.e.,

$$\mathbf{A}' = \begin{pmatrix} a_{ll} & a_{ml} \\ a_{lm} & a_{mm} \end{pmatrix}.$$

An orthogonal matrix $\mathbf{C}' = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix}$, for which $[\mathbf{C}'\mathbf{A}']_{21} = 0$ will require that the angle of rotation satisfies the equations

$$a_{ll}\sin\theta + a_{lm}\cos\theta = 0 \tag{A.29}$$

or

$$\sin\theta = \frac{-a_{lm}}{\sqrt{a_{lm}^2 + a_{ll}^2}} \quad \text{and} \quad \cos\theta = \frac{a_{ll}}{\sqrt{a_{lm}^2 + a_{ll}^2}}. \tag{A.30}$$

The solution (A.30) always exists for real matrix elements in $\mathbf{A}$, although the solution is not unique. The actual matrix $\mathbf{C}$ is made up of the elements of $\mathbf{C}'$ located exactly as the four elements in $\mathbf{A}$ together with unit elements along the remaining diagonal positions. Thus the matrix $\mathbf{C}$ used to null the lm-th element of a matrix $\mathbf{A}$ is given by

$$\mathbf{C} = \begin{pmatrix} 1 & & \cdots & & \\ & c & \cdots & & -s \\ & & 1 \quad \cdots & & \\ & & 1 & & \\ & s & \cdots & \cdots & c \\ & & \cdots & \cdots & & 1 \end{pmatrix}, \tag{A.31}$$

where the cosine and sine elements are located along the m-th and the l-th rows. The QR decomposition theorem can now be stated.

Theorem 11: Any square matrix **A** defined over the field of real numbers can be decomposed so that

$$\mathbf{A} = \mathbf{QR},$$

where **Q** is an orthogonal matrix and **R** is an upper triangular matrix. In particular, **Q** can be constructed using a series of Givens rotations, $\mathbf{C}_1, \mathbf{C}_2, \ldots, \mathbf{C}_N$ to null all the elements below the main diagonal. Then **Q** is given by

$$\mathbf{Q} = (\mathbf{C}_N \cdots \mathbf{C}_2\mathbf{C}_1)^{\mathrm{T}} \tag{A.32}$$

and **R** is given by

$$\mathbf{R} = (\mathbf{C}_N \cdots \mathbf{C}_2\mathbf{C}_1)\mathbf{A}. \tag{A.33}$$

It is no exaggeration to say that LU and QR decompositions form the backbone of the body of computational techniques in numerical matrix algebra.

As **Q** is an orthogonal matrix it is immediately evident that **A** is similar to a matrix **B** defined by the factors **Q** and **R** in the product

$$\mathbf{B} = \mathbf{RQ} = \mathbf{Q}^{-1}\mathbf{QRQ} = \mathbf{Q}^{-1}\mathbf{AQ} = \mathbf{Q}^{\mathrm{T}}\mathbf{AQ}. \tag{A.34}$$

Hence, **A** and **B** have the same eigenvalues. Intuitively, **B** is obtained from **A** by applying Givens rotations to the rows as well as to the columns. This has the effect of decreasing the magnitudes of the off-diagonal elements. Suppose the matrix **B** is QR decomposed, so that

$$\mathbf{B} = \mathbf{Q}_1\mathbf{R}_1.$$

Then, the matrix $\mathbf{B}_1 = \mathbf{R}_1\mathbf{Q}_1$ will still be similar to **A**. But, since it is obtained by applying the Givens rotations to the rows and columns of **B**, its off-diagonal elements will be even smaller in magnitude than those in **B**. One sees that by iterating this procedure a diagonal matrix consisting of eigenvalues of the original matrix can be obtained eventually. The next example illustrates this procedure of estimating the eigenvalues of a symmetric matrix.

Example 27: Find the eigenvalues of the matrix $\mathbf{A} = \begin{pmatrix} 4 & 2 & 4 \\ 2 & 10 & 5 \\ 4 & 5 & 21 \end{pmatrix}$.

Step 1: Perform a QR decomposition for **A** so that

$$\mathbf{A} = \mathbf{QR}, \quad \text{where } \mathbf{Q} = \begin{pmatrix} 0.6667 & 0.4134 & 0.6202 \\ 0.3333 & -0.9096 & 0.2481 \\ 0.6667 & 0.0413 & -0.7442 \end{pmatrix}$$

and

$$\mathbf{R} = \begin{pmatrix} 6 & 8 & \frac{55}{3} \\ 0 & -\sqrt{65} & \frac{-49}{\sqrt{195}} \\ 0 & 0 & \frac{-473}{\sqrt{325}} \end{pmatrix} = \begin{pmatrix} 6.000 & 8.000 & 18.333 \\ 0 & -8.0623 & -2.0259 \\ 0 & 0 & -11.9073 \end{pmatrix}.$$

Step 2: Find the matrix $\mathbf{B} = \mathbf{RQ}$ giving

$$\mathbf{B} = \begin{pmatrix} 18.8889 & 4.0380 & -7.9384 \\ 4.0380 & 7.2496 & 0.4923 \\ -7.9382 & 0.4923 & 8.8615 \end{pmatrix}.$$

Note that $\mathbf{B}$ is symmetric as expected.

Step 3: Repeat Steps 1 and 2 for the matrix obtained in Step 2 until the off-diagonal elements are sufficiently small. For this example, after six iterations of Steps 1 and 2, the following matrix is obtained

$$\mathbf{B}_7 = \begin{pmatrix} 23.9282 & -0.0029 & -0.00 \\ -0.0029 & 8.0999 & 0.0101 \\ -0.00 & 0.0101 & 2.9719 \end{pmatrix}.$$

The diagonal elements of $\mathbf{B}_7$ are exactly the eigenvalues of the matrix $\mathbf{A}$ to four decimal places.

While LU, QR and eigenvalue decompositions are important matrix factorizations for square matrices, there are physical systems which require nonsquare matrix representations. Not the least important of these is the least squares problem, where a solution to the following $m \times n$ system is sought:

$$\mathbf{Ax} = \mathbf{b}, \quad (m > n), \tag{A.35}$$

so as to minimize the squared error defined by

$$||\mathbf{Ax}_s - \mathbf{b}||^2. \tag{A.36}$$

Here, the solution $\mathbf{x}_s$ that minimizes (A.36) is generally denoted by $\mathbf{x}_{\mathrm{LS}}$ as the l*east squares* solution to (A.35). Since $\mathbf{A}$ is a nonsquare matrix, the notion of a *pseudoinverse* for $\mathbf{A}$ has to be introduced. If n is the rank of matrix $\mathbf{A}$ (i.e., the columns of $\mathbf{A}$ are linearly independent), this pseudoinverse is defined as

$$\mathbf{A}^{\mathrm{I}} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}$$

from which the solution $\mathbf{x}_{\mathrm{LS}} = \mathbf{A}^{\mathrm{I}}\mathbf{b} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$ is obtained. The existence of this pseudoinverse is assured by the invertibility of the matrix $(\mathbf{A}^{\mathrm{T}}\mathbf{A})$, which is full rank (rank $= n$).

However, when r, the rank of $\mathbf{A}$, is less than n, the matrix $(\mathbf{A}^{\mathrm{T}}\mathbf{A})$ will be rank deficient and a more general pseudoinverse has to be defined. Such a pseudoinverse depends on another important orthogonal decomposition of the matrix $\mathbf{A}$ called the *singular value decomposition* or SVD. The decomposition is described formally by the SVD theorem.

Theorem 12: For a real $m \times n$ matrix A, there exist orthogonal matrices $\mathbf{U}$ and $\mathbf{V}$ of dimension $m \times m$ and $n \times n$, respectively, such that

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}, \tag{A.37}$$

where $\Sigma = \begin{pmatrix} \mathbf{D} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ is an $m \times n$ matrix in which the submatrix $\mathbf{D} = \mathrm{diag}(\sigma_1, \sigma_2, \sigma_3, \ldots, \sigma_r)$ is a diagonal matrix with strictly positive elements σ_i, known as the singular values of $\mathbf{A}$ and $r \leq \min(m, n)$ is the rank of $\mathbf{A}$.

While the SVD may seem unfamiliar at this point, it can be related readily to the eigenvalue decomposition of a matrix. From (A.37) it can be seen that

$$\mathbf{A}^{\mathrm{T}}\mathbf{A} = \mathbf{V}(\Sigma^{\mathrm{T}}\Sigma)\mathbf{V}^{\mathrm{T}} \quad \text{and} \quad \mathbf{A}\mathbf{A}^{\mathrm{T}} = \mathbf{U}^{\mathrm{T}}(\Sigma\Sigma^{\mathrm{T}})\mathbf{U}, \tag{A.38}$$

using the unitarity property of $\mathbf{U}$ and $\mathbf{V}$ (i.e., $\mathbf{U}^{\mathrm{T}}\mathbf{U} = \mathbf{I}_m$, $\mathbf{V}\mathbf{V}^{\mathrm{T}} = \mathbf{I}_n$). Equation (A.38) is easily seen as showing the similarity transformation of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ to the diagonal matrices $\Sigma^{\mathrm{T}}\Sigma$ and $\Sigma\Sigma^{\mathrm{T}}$, respectively. Since these latter matrices are of the form diag $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_r^2, 0, \ldots)$, (A.38) represents the eigenvalue decompositions of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, while the columns of $\mathbf{V}$ and $\mathbf{U}$ are related to the eigenvectors of these matrices. The column vectors $\mathbf{u}_i$ and $\mathbf{v}_i$, $i = 1, 2, \ldots, r$ are respectively called the left and the right eigenvectors of the matrix $\mathbf{A}$. We show two examples here on the SVD, one on a full rank square matrix and one on a rank deficient nonsquare matrix based on its relation to the eigenvalue decomposition as shown by (A.38).

Example 28: Find the SVD for the matrix $\mathbf{A} = \begin{pmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{pmatrix}$.

Form the product $\mathbf{A}^{\mathrm{T}}\mathbf{A} = \begin{pmatrix} 6.12 & 3.84 \\ 3.84 & 3.88 \end{pmatrix}$ and the product $\mathbf{A}\mathbf{A}^{\mathrm{T}} = \begin{pmatrix} 3.88 & 3.84 \\ 3.84 & 6.12 \end{pmatrix}$.

The eigenvalue decomposition of these two matrices gives $\lambda_1 = 9$ and $\lambda_2 = 1$. The eigenvectors for $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ are $\mathbf{v}_1 = (0.8 \ \ 0.6)^{\mathrm{T}}$, and $\mathbf{v}_2 = (-0.6 \ \ 0.8)^{\mathrm{T}}$ and those for $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ are $\mathbf{u}_1 = (0.6 \ 0.8)^{\mathrm{T}}$ and $\mathbf{u}_2 = (0.8 \ -0.6)^{\mathrm{T}}$. From this is obtained the SVD for $\mathbf{A}$ as

$$\mathbf{A} = \begin{pmatrix} 0.96 & 1.72 \\ 2.28 & 0.96 \end{pmatrix} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}} = \begin{pmatrix} 0.6 & 0.8 \\ 0.8 & -0.6 \end{pmatrix} \begin{pmatrix} 3 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.8 & 0.6 \\ -0.6 & 0.8 \end{pmatrix}.$$

Note that the singular values of $\mathbf{A}$ are the positive square roots of the eigenvalues of $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ and $\mathbf{A}\mathbf{A}^{\mathrm{T}}$.

Example 29: Find the SVD of the matrix $\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 2 & 4 & 6 & 8 \end{pmatrix}$.

Note that this is a rank 2 nonsquare matrix, meaning that both the matrices $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$ are singular with at least one zero eigenvalue.

For the matrix $\mathbf{A}^{\mathrm{T}}\mathbf{A}$, the eigenvalues are 176.7179, 3.2821, 0 and 0. The corresponding eigenvectors are

$$\mathbf{v}_1 = (0.2138, 0.3389, 0.5822, 0.7074)^{\mathrm{T}},$$

$$\mathbf{v}_2 = (-0.5309, 0.4612, -0.5774, 0.4147)^{\mathrm{T}},$$

$$\mathbf{v}_3 = (0.4838, -0.5159, -0.4930, 0.5067)^{\mathrm{T}}$$

and $\mathbf{v}_4 = (0.6621, 0.6374, -0.2908, -0.2661)^{\mathrm{T}}$.

For the matrix $\mathbf{A}\mathbf{A}^{\mathrm{T}}$, the eigenvalues are 176.7179, 3.2821 and 0. The corresponding eigenvectors are

$$\mathbf{u}_1 = (0.4113, 0.3925, 0.8227)^{\mathrm{T}},$$

$$\mathbf{u}_2 = (0.1755, -0.9198, 0.3511)^{\mathrm{T}}$$

and $\mathbf{u}_3 = (-0.8944, 0, 0.4472)^{\mathrm{T}}$.

Combining these the SVD of $\mathbf{A}$ can be obtained as

$$\begin{aligned} \mathbf{A} = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 3 \\ 2 & 4 & 6 & 8 \end{pmatrix} &= \mathbf{U}\Sigma\,\mathbf{V}^{\mathrm{T}} \\ &= \begin{pmatrix} 0.4113 & 0.1755 & -0.8944 \\ 0.3925 & -0.9198 & 0 \\ 0.8227 & 0.3511 & 0.4472 \end{pmatrix} \begin{pmatrix} 13.2935 & 0 & 0 & 0 \\ 0 & 1.8116 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix} \\ &\quad \times \begin{pmatrix} 0.2138 & 0.3389 & 0.5822 & 0.7074 \\ -0.5309 & 0.4612 & -0.5774 & 0.4147 \\ 0.4838 & -0.5159 & -0.4930 & 0.5067 \\ 0.6621 & 0.6374 & -0.2908 & -0.2661 \end{pmatrix} \end{aligned}$$

Again note that the singular values are the positive square roots of the eigenvalues of $\mathbf{A}\mathbf{A}^{\mathrm{T}}$ and $\mathbf{A}^{\mathrm{T}}\mathbf{A}$. It should also be pointed out that while the eigenvectors of nonzero eigenvalues (both right and left) are determined to a sign, those of zero eigenvalues are required simply to be orthogonal to those of the nonzero eigenvalues. Thus, for example, $\mathbf{v}_3$ and $\mathbf{v}_4$ are required to be orthogonal to $\mathbf{v}_1$ and $\mathbf{v}_2$ and also be mutually orthogonal. They may therefore be quite different from the ones shown in the above example.

Returning to the solution of the rank deficient least squares problem of (A.35), the more general pseudoinverse of the matrix $\mathbf{A}$ can now be defined as

$$\mathbf{A}^{\mathrm{I}} = \mathbf{V}\Sigma^{+}\mathbf{U}^{\mathrm{T}}, \tag{A.39}$$

where $\Sigma^{+} = \begin{pmatrix} \mathbf{D}^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{pmatrix}$ and $\mathbf{D}^{-1} = \mathrm{diag}(\sigma_1^{-1}, \sigma_2^{-1}, \ldots, \sigma_r^{-1})$. Applying this to the solution of (A.35) gives $\mathbf{x}_{\mathrm{LS}} = \mathbf{A}^{\mathrm{I}}\mathbf{b}$ and the minimized squared error is then $||\,(\mathbf{A}\mathbf{A}^{\mathrm{I}} - \mathbf{I})\mathbf{b}||^2$.

Aside from providing the pseudoinverse and thus the least squares solution for a rank deficient nonsquare system such as (A.35) there are important properties associated with the SVD of a matrix $\mathbf{A}$. This section is concluded with a list of some of these important properties.

Properties of SVD of an $m \times n$ matrix A: If $\mathbf{U}^{\mathrm{T}}\mathbf{A}\mathbf{V} = \mathrm{diag}(\sigma_1, \sigma_2, \ldots, \sigma_p)$, where $p = \min(m, n)$, and $\sigma_i = 0$, for $i > r$, then

1. Rank $(\mathbf{A}) = r$.
2. The null space of $\mathbf{A}$ is $\mathrm{N}(\mathbf{A}) = \mathrm{span}\,\{\mathbf{v}_{r+1}, \mathbf{v}_{r+2}, \ldots, \mathbf{v}_n\}$.
3. The range space of A is $\mathrm{R}(\mathbf{A}) = \mathrm{span}\,\{\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_r\}$.
4. $\mathbf{A} = \sum_{i=1}^{r} \sigma_i \mathbf{u}_i \mathbf{v}_i^{\mathrm{T}}$
5. The Frobenius norm of $\mathbf{A}$ is $||\,\mathbf{A}\,||_{\mathrm{F}}^2 = \sum_{i=1}^{r} \sigma_i^2$.
6. The 2-norm of $\mathbf{A}$ is $||\,\mathbf{A}\,||^2 = \sigma_1^2$.

Problems and Exercises A.3

1. Find the LU decomposition for the following matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 8 \\ 3 & 8 & 14 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 5 & 4 & 1 \\ 10 & 9 & 4 \\ 10 & 13 & 15 \end{pmatrix}.$$

2. Find the Doolittle and Crout factorizations of the following matrices:

$$\mathbf{A} = \begin{pmatrix} 3 & 9 & 6 \\ 18 & 48 & 39 \\ 9 & -27 & 42 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 2 & 2 & 4 \\ 4 & 5 & 13 \\ 10 & 14 & 43 \end{pmatrix}.$$

3. Show that the matrix $\mathbf{A}$ in Exercise 1 is symmetric positive definite and find its Cholesky factorization.

4. The LU decomposition for a nonsingular matrix **A** is based on Gauss eliminations of matrix elements below the main diagonal. Explain why the matrix $\mathbf{G}^{-1}$ representing the inverse of the Gauss eliminations can be permuted to give the required lower triangular form.

5. Using the Cholesky decomposition in Exercise 3, solve the matrix equation

$$\begin{pmatrix} 1 & 2 & 3 \\ 2 & 5 & 8 \\ 3 & 8 & 14 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} 4 \\ 3 \\ 1 \end{pmatrix}.$$

6. Explain why by requiring the diagonal elements of **L** or of **U** to be ones, that the LU decomposition of a given nonsingular matrix will be unique.

7. Determine the QR decomposition of the following matrices:

$$\mathbf{A} = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \\ 3 & 4 & 4 \end{pmatrix} \quad \text{and} \quad \mathbf{B} = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 5 & 4 \\ 1 & 4 & 7 \end{pmatrix}.$$

8. If **A** is a square singular matrix defined over the field of real numbers $\boldsymbol{R}$, determine whether there exists a QR decomposition for **A**.

9. Find the eigenvalues of matrices **A** and **B** in Exercise 7 using the QR decomposition procedure described in this section.

10. Examine the procedure used in Exercise 9 to obtain the eigenvalues. Can the corresponding eigenvectors be extracted from the factor matrices (**Q**'s and **R**'s) without directly solving the simultaneous equations? (**Hint**: the final diagonal matrix is similar to the original matrix through the set of **Q** (orthogonal) matrix factors.)

A.4 Signal and its representations

We conclude this Appendix with a discussion of signal and its representations. What is a signal and why is it important to us? The second part of this question is probably more easily answered. When we look, we see words or images; when we touch, we feel heat or cold and when we listen, we hear music or noise. The words or images, heat or cold and music or noise are all perceived by our senses, transmitted to our brains where they are processed and understood. For want of a better word, what is being perceived, transmitted and processed is called a "signal". In this wide sense, it would be impossible to imagine a world with no signals.

So, in a very intuitive way, signals are manifestations of physical phenomena; manifestations that are sufficiently concrete so they can be perceived and understood. As the phenomenon changes, so does the associated signal. Since most changes are observed as time varies, it is natural to think about signals as functions of time. For example, the location of an eagle taking flight is a continuous function of time while the Dow–Jones average changes from day to day and is a function of discrete time. Signal can also vary

as a function of location. In short, a signal is just a function of some independent variable or variables. The mathematical definition of a function need not be strictly adhered to in this definition of a signal.

To understand signals in a more technical sense, they need to be classified and there are many ways of doing that, depending on what processing may be desirable.

Deterministic or random signals: A signal can be deterministic or random depending on the physical process it represents. Suppose that

$$f = f(t) \tag{A.40}$$

is a continuous signal or function of time t. When $f(t)$ can be determined as t changes from t_1 to t_2, the signal is regarded as deterministic. Generally, the function f will satisfy a differential equation of some kind, which together with prescribed initial conditions will completely determine f. Hence, the voltage of an RC circuit is completely determined by the values of the elements, the initial voltage and current, and the second-order differential equation derived from Kirchoff's law.

On the other hand, when f is governed by a random process and is not predictable, the signal is random. The Dow–Jones daily average is a good example. The processing of this kind of signal is usually done with statistical techniques.

While a predictable signal implies that it is deterministic, the reverse is not necessarily true. There are chaotic signals generated by deterministic dynamical systems that appear to be random. Weather patterns are a prime example in which short-term predictions are fairly accurate while long-term accurate predications are impossible to make. Thus, although the variables governing the weather such as temperature and pressure satisfy a set of dynamical equations, the solutions can be chaotic and can appear to be random. In this work chaotic signals are not dealt with.

Energy or power signals: A signal can also be classified according to its energy or power content. Define energy E and power P for a signal $f(t)$ as

$$E = \lim_{T\to\infty} \int_{-T}^{T} f^2(t)\,\mathrm{d}t \tag{A.41}$$

and

$$P = \lim_{T\to\infty} \frac{1}{2T} \int_{-T}^{T} f^2(t)\,\mathrm{d}t. \tag{A.42}$$

Definition 10: $f(t)$ is an energy signal if $0 < E < \infty$ and $P = 0$; it is a power signal if $E \to \infty$ and $0 < P < \infty$. A signal is not realistic if it is neither a power signal nor an energy signal.

A time signal can also be represented as a frequency signal through a valid transform. When the energy or power of such a signal is expressed in the frequency domain, we have

the notion of energy or power spectra. For the $f(t)$ in (A.40), if

$$E = \int_{-\infty}^{\infty} G(\omega)\,\mathrm{d}\omega$$

or

$$P = \int_{-\infty}^{\infty} S(\omega)\,\mathrm{d}\omega, \tag{A.43}$$

then $G(\omega)$ and $S(\omega)$ are respectively the energy density and power density spectra of f.

Periodic or aperiodic signals: Another classification which is of significance is whether or not a signal is periodic. If

$$f(t+T) = f(t), \quad t \in \boldsymbol{R} \tag{A.44}$$

for some finite T, then f is a periodic function or signal. When T is the minimum for which (A.44) is true it is called the *period*. If (A.44) is not satisfied for any nonzero T, f is said to be *aperiodic*.

Periodic signals form a very important class. Much of classical transform analysis is based on the notion of periodic signals of different periods. It is interesting to note that a nonsingular periodic signal is naturally a power signal. One of the most important ways of representing a periodic signal is by using complex numbers. For example,

$$f(t) = A\exp(\mathrm{j}\omega t), \tag{A.45}$$

where A and ω are real and fixed, is a periodic signal with a period of $T = 2\pi/\omega$. The real and imaginary parts of (A.45) are respectively given by

$$\mathrm{Re}[f(t)] = A\cos(\omega t), \quad \mathrm{Im}[f(t)] = A\sin(\omega t), \tag{A.46}$$

both of which are periodic. These are sometimes referred to as the I and Q channels of the signal f.

Continuous or discrete signals: If the signal in (A.40) is defined for all $t \in \boldsymbol{R}$, $f(t)$ is defined as a continuous signal. If f is defined only for $t \in \{t_1, t_2, \ldots, t_N\}$, with $t_i \in \boldsymbol{R}$, $f(t)$ is said to be a discrete signal. In particular, if

$$f(t_i) = f(t) \quad \text{for } t = t_i, \;\; i = 1, 2, \ldots, N, \tag{A.47}$$

$f(t_i)$ is said to be the sampled version of $f(t)$ and t_i's are the sampling instants.

Discrete signals need not in any way be related to a continuous signal through sampling. Naturally occurring discrete signals such as daily maximum temperature, monthly utility

bills or annual rainfalls abound, either as a result of the measurement process or as a result of the underlying physical phenomena. However, practically all digital signal processing techniques are developed for sampled signals which are by definition discrete.

While a discrete signal is defined at only given instants of time (or whatever the independent variable happens to be) its values need not be discrete. When the signal can only assume discrete values, it is said to be quantized. Digital signals are generally understood to be both quantized and discrete. Thus,

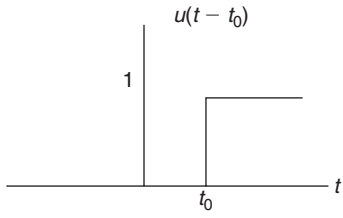
$$f = f(t) \text{ for which } f \in \{f_1, f_2, \ldots, f_M\} \quad \text{and} \quad t \in \{t_1, t_2, \ldots, t_N\} \tag{A.48}$$

is a discrete signal with M levels of quantization and is typical of a digital signal.

We have seen in the above discussion on the classification of signals that a signal is mathematically represented by a function. However, the notion of function here is not strictly mathematical. Many signals are well presented by distributions, particularly when special properties of the signal under study have to be highlighted. Before we examine some typical operations on signals such as sampling, interpolation, etc., we list examples of some elementary signals and their properties which will be of use.

Example 30: The Heaviside or unit step signal.

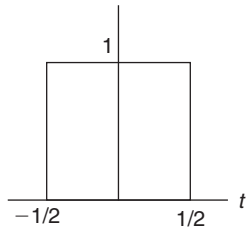
$$\begin{aligned} u(t - t_0) &= 0 \quad \text{for } t \leq t_0, \\ &= 1 \quad \text{for } t > t_0. \end{aligned} \tag{A.49}$$



It is easy to see that $u(t - t_0)$ is a deterministic aperiodic continuous power signal.

Example 31: The unit pulse signal.

$$\pi(t) = u\left(t + \frac{1}{2}\right) - u\left(t - \frac{1}{2}\right). \tag{A.50}$$

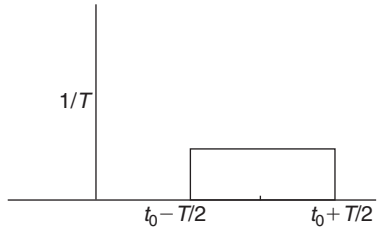


Note that $\pi(t)$ is made up of two unit step signals. It is easy to see that $\pi(t)$ is a deterministic aperiodic continuous energy signal.

In both examples, the word continuous is used not to imply mathematical continuity but to distinguish it from discrete signals. As can be easily seen, both $u(t)$ and $\pi(t)$ contain jump discontinuities at which the functions cannot be differentiated.

Example 32: The scaled rectangular pulse signal.

$$\pi_T(t - t_0) = \left(\frac{1}{T}\right)\left\{u\left[t + \left(t_0 - \frac{T}{2}\right)\right] - u\left[t - \left(t_0 + \frac{T}{2}\right)\right]\right\}. \tag{A.51}$$



$\pi_T(t - t_0)$ is a deterministic continuous aperiodic energy signal. Since $\pi_T(t - t_0)$ is scaled, its energy content is kept at unity, independent of the extent of the pulse width T.

Example 33: The unit impulse signal (Dirac's delta function).

$$\delta(t - t_0) = \lim_{T \to 0} \pi_T(t - t_0). \tag{A.52}$$

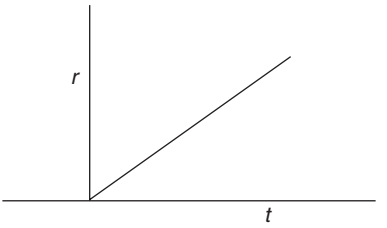
This is not a function in the ordinary sense. Its value is zero everywhere except at $t = t_0$, where it is not defined. The integral of this function, as can be seen by examining the integral of $\pi_T(t - t_0)$ for finite T and taking the limit of T to zero, is always unity. In addition, it can be shown that for any function $f(t)$, defined at $t = t_0$ the following equality holds

$$f(t_0) = \int_{-\infty}^{\infty} f(t)\delta(t - t_0)\,\mathrm{d}t. \tag{A.53}$$

Hence, $\delta(t - t_0)$ is a deterministic continuous aperiodic energy signal.

Example 34: The infinite ramp signal.

$$\begin{aligned} r(t) &= 0 \quad \text{for } t < 0, \\ &= t \quad \text{for } t \geq 0. \end{aligned} \tag{A.54}$$



We note that this cannot be a physical signal since it is neither a power nor an energy signal. It is however deterministic, continuous (even in the mathematical sense) and aperiodic.

Example 35: The unit ramp function.

$$r_T(t) = \left(\frac{2}{T}\right) r(t)\pi_T(t). \tag{A.55}$$

By combining $r(t)$ with the scaled rectangular pulse, $r_T(t)$ is obtained. It is as can be easily seen an energy signal.

In many ways, the Examples 30–35 may be regarded as elemental signals from which more complex ones can be constructed. In fact, Examples 31 and 35 are just such composite signals. Example 33 is a particularly important elemental signal in that it can be used as a "probe" to sample a general continuous signal at discrete times. We list two other examples involving periodic signals.

Example 36: Single frequency sinusoid.

$$s(t) = A\cos(\omega t + \theta), \tag{A.56}$$

where A, ω and θ are respectively the amplitude, frequency and phase of the sinusoid. The signal is deterministic, continuous and periodic; it is obviously a power signal.

Example 37: Linear combination of sinusoids.

$$S(t) = \sum_{i=1}^{N} s_i(t) = \sum_{i=1}^{N} A_i \cos(\omega_i t + \theta_i), \tag{A.57}$$

where A_i, ω_i and θ_i are the amplitude, frequency and phase of the individual sinusoids. While the signal in (A.57) is similar to $s(t)$ in (A.56) in all its classifications, it may however not be a periodic signal. Equation (A.57) is a special case of the more general Fourier series analysis which forms the foundation of many classical discrete transforms.

As a final example, we demonstrate the operation of sampling on a continuous signal $f(t)$, using the probe or sampling ability of the unit impulse signal or the Dirac delta function.

Example 38: Let $f(t)$ be a continuous signal to be sampled every T_s (the sampling interval). The operation of sampling means that the discrete signal $f^*(t)$ will be

$$\begin{aligned} f^*(t) &= f(t_k) \quad \text{for } t = t_k = kT_s \\ &= 0 \qquad \text{otherwise.} \end{aligned} \tag{A.58a}$$

By using the property of the Dirac delta function we have for causal signals

$$\begin{aligned} f^*(t) &= \int_0^\infty f(\tau)\delta(\tau - t_k)\,\mathrm{d}\tau \quad \text{for } t = t_k \\ &= 0 \quad \text{otherwise.} \end{aligned} \tag{A.58b}$$

This can be more compactly represented if the Kronecker delta function is introduced

$$\begin{aligned} \delta_{nk} &= 1 \quad \text{if } n = k, \\ &= 0 \quad \text{otherwise.} \end{aligned} \tag{A.59}$$

The sampled signal can now be written as

$$f^*(t) = \sum_k f(t_k)\delta_{tt_k} = \sum_k \int_0^\infty f(\tau)\delta(\tau - t_k)\,\mathrm{d}\tau\delta_{tt_k}. \tag{A.60}$$

It is perhaps worth noting that (A.60) is very often expressed incorrectly as

$$f^*(t) = \sum_k f(t_k)\delta(t - t_k),$$

using the Dirac delta function instead of the Kronecker delta function.

While a continuous signal can be sampled at discrete times, a discrete signal can be interpolated so that a continuous signal will be reconstructed as a result. This possibility is stated most succinctly in the well-known sampling theorem by Shannon, which we now use to conclude this section on signal and its representation.

Sampling theorem (Shannon, 1949): If $f(t)$, the signal to be sampled is band limited, i.e., if it contains no frequency components above some f_c – then it can be completely described (reconstructed) by uniformly spaced samples taken at a rate of at least $2f_c$ samples/s.

This page intentionally left blank

Index