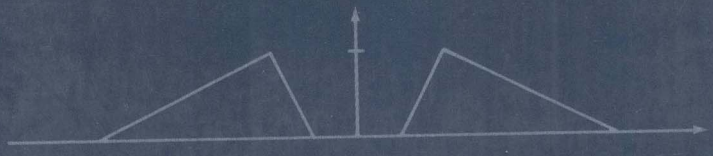
Georgia Tech

DIGITAL SIGNAL PROCESSING
LABORATORY SERIES

DIGITAL FILTERING

A Computer Laboratory Textbook



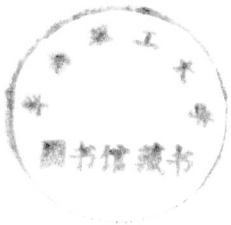
Russell M. Mersereau
Mark J.T. Smith

Georgia Tech
DIGITAL SIGNAL PROCESSING
LABORATORY SERIES

DIGITAL FILTERING
A Computer Laboratory Textbook

RUSSELL M. MERSEREAU
MARK J. T. SMITH

Georgia Institute of Technology



JOHN WILEY & SONS, INC.
New York • Chichester • Brisbane • Toronto • Singapore

附软盘1件

Acquisitions Editor	Steven Elliot
Marketing Manager	Debra Riegert
Senior Production Supervisor	Savoula Amanatidis
Cover Designer	Bonnie Cabot
Illustration Coordinator	Sigmund Malinowski
Manufacturing Manager	Andrea Price

This book was typeset in Times Roman by the authors and printed and bound by Malloy Lithographing, Inc. The cover was printed by Phoenix Color Corp.

Library of Congress Cataloging in Publication Data:

Mersereau, Russell M.
Digital filtering : a computer laboratory textbook / Russell M. Mersereau, Mark J.T. Smith.
p. cm.
System requirements for computer disk: IBM-compatible PC (80286 microprocessor or better recommended); 640K RAM; MS-DOS; hard disk; CGA, EGA, or VGA display; floating point math co-processor and standard ASCII text editor recommended.
Includes bibliographical references (Pref.).
Includes index.
ISBN 0–471–51694–5
Electric filters, Digital–Design and construction–Data processing. 2. Signal processing–Digital techniques–Data processing. 3. Digital filters (Mathematics) I. Smith, Mark J. T. II. Title.
TK7872.F5M467 1993
621.3815′324′078–dc20 93–17580
CIP

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

Printed and bound by Malloy Lithographing, Inc.

DIGITAL FILTERING
A Computer Laboratory Textbook

THE GEORGIA TECH DIGITAL SIGNAL PROCESSING LABORATORY SERIES

Editors:

Thomas P. Barnwell III

Monson Hayes

Russell M. Mersereau

Mark J. T. Smith

Texts in this series include:

Introduction to Digital Signal Processing: A Computer Laboratory Textbook
by Mark J. T. Smith and Russell M. Mersereau

Digital Filtering: A Computer Laboratory Textbook
by Russell M. Mersereau and Mark J. T. Smith

Spectral Analysis: A Computer Laboratory Textbook
by Monson Hayes

Speech Coding: A Computer Laboratory Textbook
by Thomas P. Barnwell III, Kambiz Nayebi, and Craig H. Richardson

Foreword

After spending decades in the research laboratory, digital signal processing (DSP) is now emerging to make a significant impact on many areas of technology. As a result, DSP is becoming a basic subject in the electrical engineering curriculum. Although numerous textbooks and reference books are available to present the theory and applications of DSP, few of these books provide much in the way of "hands-on" experience that can help a student translate equations and algorithms into insight.

Experience during the past fifteen years at the Georgia Institute of Technology in using computers with both basic and advanced courses in DSP has shown that the personal computer can be an extremely effective learning aid when it is combined with well-designed exercises and effective software support. The Georgia Tech Digital Signal Processing Laboratory Series builds on this teaching experience to provide a set of computer laboratory books that can be used either to supplement traditional classroom/textbook presentations of the subject or as a self-study aid.

The value of computer-based laboratory experience is clear. However, just what this experience should be is somewhat dependent on the computer resources available and on the computer skills of the students. The following three approaches have proved to be effective:

1. Provide the student with a program or set of programs that perform specific DSP functions. In this situation, exercises are necessarily limited to running the programs on test data and observing the results.
2. Provide the student with a set of exercises that can be carried out by using a set of macros or low-level functions that can be strung together in some sort of convenient software environment. This approach has the virtue of flexibility and is much less restrictive.
3. Provide the student with test data and suggestions for projects to be carried out with

whatever programming resources are available. Clearly, this is the least restrictive approach, but is the most demanding of the student's programming/computer skills.

The first approach is likely to be frustratingly limited for students who are learning fundamental concepts, but it is very appropriate when the goal is to demonstrate complex algorithms that would require a great deal of time if students were to implement them on their own. For example, digital speech processing systems often combine many basic DSP functions and often have many parameters whose effects can only be illustrated and studied by using an elaborate program. Another example is filter design, where students can learn the properties of different approximation methods by simply applying those methods to the same set of specifications. At the opposite extreme is the third approach, which is obviously most suited for advanced courses or independent study where appropriate computer programming skills can be required. The second approach is perhaps the best compromise for developing insight into the fundamental algorithms and concepts of DSP. The book *Digital Filtering: A Computer Laboratory Textbook,* is based on primarily this approach and is the second book in the Georgia Tech Digital Signal Processing Laboratory Series. It addresses the set of topics related to filter design, implementation, and analysis and is a follow-on to the first book *Introduction to Digital Signal Processing: A Computer Laboratory Textbook.*

Digital Filtering includes more than 170 exercises that can be carried out under DOS using carefully designed software provided with the book. This software has a wide range of basic operations, a large set of filter design functions, and a structure that allows these functions to be strung together to perform more complex functions. At Georgia Tech, this computer laboratory mode of operation has been underway for several years. Student response to both the software and the exercises has been extremely favorable. Students appreciate the ease with which they can begin to actually do something with what they are learning in the classroom.

There is no doubt that DSP education is moving toward the greater use of computers. Indeed, few subjects in the electrical engineering curriculum are so well suited to the use of computers in instruction. The Georgia Tech Digital Signal Processing Laboratory Series, whose authors have many years experience in teaching and research in the DSP field, is a valuable contribution to this emerging trend in electrical engineering education.

Ronald W. Schafer
John O. McCarty Institute Professor
Georgia Institute of Technology

Preface

Filter design is an important topic in the area of discrete-time processing. There are many digital signal processing textbooks that present a good discussion of the theory, provide a variety of illustrative examples, and include a wide selection of homework problems related to the major topics. The purpose of this book, however, is somewhat different. It is to provide hands-on exposure to digital filter design in a computer environment. It can be used to complement a digital signal processing (DSP) text, as the text for an introductory *laboratory* course in digital signal processing, or as a self-paced introduction to DSP basics.

The book includes a library of DSP computer functions that run on personal computers using the DOS operating system. The philosophy underlying this text is to provide the DSP newcomer with the experience of working with complex design formulas and design algorithms without having to write and debug large programs. Computer-based exercises have been a very important component in the digital signal processing course offerings at Georgia Tech and strongly contribute to an enriched understanding of the material.

This book is the second in a two-part sequence that focuses on the fundamental concepts of digital signal processing. The first book, *Introduction to Digital Signal Processing: A Computer Laboratory Textbook*, covers linear systems, the discrete Fourier transform, sampling, the z-transform, the DFT and FFT, and certain other topics. This text is a continuation and is devoted to digital filters, digital filter design, and filter implementation. It assumes that the reader is familiar with most of the topics covered in the first text. Each chapter begins with a brief summary of the fundamentals on its topic. These discussions are followed by a set of illustrative exercises that provide a mix of theoretical, experimental, and design problems. Many of the problems are straightforward, and their solutions can be verified easily and quickly by using the computer. The exercises also include a number of more difficult problems to challenge the learner. Certain exercises can be selectively omitted without a loss of understanding, according

to the reader's level of familiarity with and interest in a particular topic.

The text is organized so that proceeding through the initial chapters and exercises in order provides a smooth introduction to the software that is used throughout the text. The presentation is most effective when chapters are selected in order but chapters may be selectively omitted without loss of continuity. It is suggested, however, that Chapter 1 be reviewed first. It reviews the fundamentals of FIR and IIR filters, establishes the notation, and provides an introduction to the software. Chapter 2 treats FIR filter design. It includes discussions and exercises on window design, the Remez exchange algorithm, and the Parks–McClellan algorithm. Chapter 3 introduces classical analog filter design methods, analog frequency translation, a variety of analog to digital filter conversion methods, and digital frequency transformations. Chapter 4 discusses allpass filters, their properties, and some fundamentals of multirate filtering. Chapter 5 is devoted to filter structures and implementation issues. Coefficient quantization, roundoff errors, and limit cycles are treated in some detail. The final chapter of the book is a collection of projects. These use the concepts and techniques discussed throughout the text to solve specific problems. A topical reference chart is included in this preface that shows how the chapters in this book and its predecessor would best complement those in several commonly used DSP textbooks.

The software used in this book is included on the enclosed disk. There are three basic executable programs: **f.exe**, which contains a diverse set of elementary signal processing and filter design functions; **g.exe**, which contains all of the display graphics functions; and **helpf.exe**, the on-line help for the **f** and **g** functions. The software will generally run on any personal computer that supports the DOS. It has been primarily tested on IBM PS/2 and HP Vectra PC systems. It is strongly recommended that the computer supporting the DSP software have a floating-point coprocessor. The programs will also run more efficiently if all of the software and data files reside on a hard disk. The graphics functions should operate properly on most EGA-, CGA-, and VGA-equipped machines.

In the development of this laboratory text we were fortunate to have received valuable feedback from colleagues and students at Georgia Tech in the United States and Georgia Tech Lorraine in France. We gratefully acknowledge Prof. James McClellan for his contributions to Chapter 6 and Dr. Steven L. Eddins for his development of the early core set of computer programs that evolved into the present software. The software has undergone much revision and modification during the course of its development. We are indebted to Mr. Faouzi Kossentini and Mr. Wilson Chung who over the last few years have revised, maintained and expanded the software as the book developed. We would like to gratefully acknowledge the following reviewers for their suggestions on changes and improvements in the manuscript: Matt Yeldin, University of California, Berkeley; Ernie Baxa, Clemson University; Enrico Del Re, Universita Delgi Studi Di Firenze; Frederick J. Harris, San Diego State University; and Janet Slocum, Tufts University. Finally, we would like to thank our wives Martha Mersereau and Cynthia Y. Smith and our children Adam and David Mersereau and Stephen, Kevin, and Jennifer Smith for their steadfast love and support over the years.

This book and its companion were written to be laboratory texts and not to be the

primary text in a lecture course. Several popular primary texts are listed below. These can be relied upon for more complete discussions, examples, and derivations of the key results that we have only summarized. The tables indicate which chapters in the primary texts provide overlapping coverage with the chapters in these two laboratory texts.

[1] L. B. Jackson, *Digital Filters and Signal Processing,* (2nd), Kluwer Academic Publishers: Boston, 1989.

[2] R. Kuc, *Introduction to Digital Signal Processing,* McGraw-Hill: New York, 1988.

[3] L. C. Ludeman, *Fundamentals of Digital Signal Processing,* Harper and Row: New York, 1986.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall: Englewood Cliffs, NJ, 1989.

[5] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*, Prentice-Hall: Englewood Cliffs, NJ, 1975.

[6] J. G. Proakis and D. G. Manolakis, *Introduction to Digital Signal Processing*, Macmillan: New York, 1988.

[7] R. A. Roberts and C. T. Mullis, *Digital Signal Processing*, Addison-Wesley: Reading, MA, 1987.

[8] R. D. Strum and D. E. Kirk, *First Principles of Discrete Systems and Digital Signal Processing*, Addison-Wesley: Reading, MA, 1988

Intro. DSP (Smith & Mers.)	Ch.1	Ch.2	Ch.3	Ch.4	Ch.5	Ch.6
1) Jackson	—	Ch.2	Ch.4	Ch.6	Ch.3	Ch.7
2) Kuc	—	Ch.2	Ch.3	Ch.3	Ch.5	Ch.4
3) Ludeman	—	Ch.1	Ch.1	Ch.1	Ch.2	Ch.6
4) Opp. & Sch., 1989	—	Ch.2	Ch.2,5	Ch.3	Ch.4	Ch.8,9
5) Opp. & Sch., 1975	—	Ch.1	Ch.1	Ch.1	Ch.2,4	Ch.3,6
6) Proakis & Manolakis	—	Ch.2	Ch.4	Ch.1	Ch.3	Ch.9
7) Roberts & Mullis	—	Ch.2	Ch.4	Ch.4	Ch.3	Ch.4,5
8) Strum & Kirk	—	Ch.3	Ch.4	Ch.2	Ch.5,6	Ch.7,8

Dig. Filters (Mers. & Smith)	Ch. 1	Ch. 2	Ch. 3	Ch. 4	Ch. 5
1) Jackson	—	Ch. 9	Ch. 8	Ch. 13	Ch. 5, 11
2) Kuc	—	Ch. 9	Ch. 8	—	Ch. 6, 10
3) Ludeman	—	Ch. 3	Ch. 3,4	—	Ch. 6, 10
4) Opp. & Schafer, 1989	—	Ch. 7	Ch. 7	Ch. 10	Ch. 6
5) Opp. & Schafer, 1975	—	Ch. 5	Ch. 5	—	Ch. 4,8
6) Proakis & Manolakis	—	Ch. 8	Ch. 8	—	Ch. 10, 7
7) Roberts & Mullis	—	Ch. 6	Ch. 6	—	Ch. 9, 10
8) Strum & Kirk	—	Ch. 9	Ch. 10	—	Ch. 11

Contents

Introduction 1

Digital filters can be used in a wide variety of applications, including separating signals from noise, compensating for linear distortions, separating signal components that have been added together, and modeling many classes of signals. This book presents a series of computer exercises in digital filtering to acquaint you with a number of filter design techniques and methods of filter implementation. It comes with custom DSP software that will allow you to design filters, study design algorithms, and study many types of digital filters without having to write programs. Although many of the fundamentals are discussed in this book, it is not intended to be a basic text in digital signal processing or in digital filtering. Instead, its goal is to supplement such a text by providing a hands-on "computer laboratory" experience.

This chapter has two purposes; first, it reviews some basic concepts related to digital filters while establishing the notation used throughout the book. It also provides a smooth introduction to the use of the software. Chapters 2 and 3 present a series of tutorial exercises on various aspects of FIR and IIR filter design, and Chapter 4 looks at some more advanced topics concerned with allpass and multirate filters. Chapter 5 explores alternative implementations for digital filters and carefully examines quantization and overflow effects that can occur in fixed-point hardware realizations. The text concludes with a series of design projects in Chapter 6 that are more lengthy in nature and require a higher level of thought and creativity.

1.1 GETTING STARTED

In this text, signal processing operations are presented in a hands-on personal computer environment that can create and display signals with only a few commands. To get started you will need the following:

- An IBM-compatible personal computer. A computer with at least an 80286 or 80386 microprocessor is preferred, although not necessary. It is also recommended that the computer contain a floating-point co-processor. This will increase the speed of the software dramatically.
- A hard disk. Since files will be created routinely when you do the exercises, it is suggested that a few megabytes of disk be available in your working directory.
- The MS-DOS operating system or its equivalent.
- Either CGA, EGA, or VGA display capability.
- The computer should contain a minimum of 640 kbytes of random access memory. Since the DSP software uses a sizable part of this memory, avoid running other memory resident programs during your work session with the software. The presence of these programs in memory reduces the memory available to the DSP software and may cause errors.

A standard ASCII text editor is also useful. It will allow you to write macros and to edit DSP files. In Chapter 6, which is the projects chapter, and in two optional problems in Chapter 5, you are asked to write computer programs. In such cases, you will need a compiler for the programming language in which you wish to work.

A *print-screen* program that allows you to make hard copies of the graphics displayed on the screen might also be useful in some cases, but is not necessary. The DSP software does not provide the capability to print graphics outputs. You will generally be asked to draw sketches of signals that are displayed on the screen.

To begin, create a DSP directory on your hard disk and copy all of the files on the enclosed diskette into that directory. Go into the DSP directory and type **install**. The programs and files provided on the disk are stored in a compressed format on the diskette. Typing **install** uncompresses the software. It is recommended that you copy these programs and files to a backup diskette as a safeguard in case they are accidentally deleted or overwritten.

You may do all of your work in this DSP directory. However, you may find it more convenient to work in another directory, thereby keeping your working files separate from the DSP software. Such a setup can be created by modifying your search path in DOS. Your DOS manual contains detailed information about customizing the operating environment for your computer.

There are two main programs in the software that contain DSP functions: **f.exe**, which contains filter design and signal processing functions; and **g.exe**, which contains graphics and display functions. The functions in **f.exe** can be used to do simple operations such as adding, subtracting, or multiplying signals as well as to perform more complex operations such as filter design, multirate filtering, and quantization simulations. Each function in this set can be invoked by simply typing **f** followed by the function name (e.g., **f add, f subtract, f multiply**).

The functions in **g.exe** allow you display signals in the time domain, z-plane, and frequency domain. These graphics functions are invoked by typing **g** followed by the function name. A list of the basic functions for this book is provided in Table 1.1.

Table 1.1. List of DSP Software Functions.

f functions			
f abessel	**f abutter**	**f acheby1**	**f acheby2**
f adaptfir	**f add**	**f aelliptic**	**f atransform**
f bartlett	**f bilinear**	**f blackman**	**f cartesian**
f cas	**f cexp**	**f convert**	**f convolve**
f diff	**f direct1**	**f direct2**	**f divide**
f dnsample	**f dtransform**	**f eformulas**	**f extract**
f fdesign	**f fft**	**f filter**	**f gain**
f hamming	**f hanning**	**f hilbert**	**f histogram**
f ideallp	**f ifft**	**f imagpart**	**f impinv**
f kaiser	**f kalpha**	**f lccde**	**f log**
f lshift	**f mag**	**f matchedz**	**f maxflat**
f median	**f multiply**	**f nlinear**	**f obutter**
f ocheby1	**f ocheby2**	**f oelliptic**	**f par**
f phase	**f pksmcc**	**f polar**	**f qcyclic**
f quantize	**f rank**	**f realpart**	**f reverse**
f revert	**f rgen**	**f rootmult**	**f rooter**
f siggen	**f snr**	**f subtract**	**f summer**
f truncate	**f upsample**	**f zeropad**	
g functions			
g afilspec	**g afreqres**	**g apolezero**	**g dfilspec**
g dtft	**g look**	**g look2**	**g polezero**
g slook2	**g sview2**	**g view**	**g view2**

Working with the software is very simple and does not require knowing much in order to get started, but there are several things that we should point out before you begin. First, signals (or sequences) are stored in files. Their content may be examined at any time by simply printing them on the screen, i.e., by entering **type** followed by the filename. As an example, try typing **f001**, which is a file provided for you on disk. After pressing the enter key, the file content will be displayed. Observe that the first five lines of the file provide information about the signal while the numbers that follow are the sequence values. This file format is convenient for modification because the file information is self-explanatory. Using a text editor, you can change coefficient values if desired as well as add or delete coefficients. In the case of the latter, the filter length and numerator order parameters would have to be changed appropriately. As another example, consider the file **impulse**, which contains the unit sample, $\delta[n]$. It is used as an input signal in many of the exercises. Type this file to the screen and observe that it contains only one sample. The fact that the starting point is zero means that the unit impulse occurs at $n = 0$.

Second, whenever you are in doubt about what a particular function does or how to use it, simply type **helpf** for an on-line description of the functions in **f.exe** and **g.exe**. Try typing **helpf** now. It will display a list of all the functions available. To obtain detailed information about a particular function listed, type **helpf** followed by the function name.

Try typing **helpf add** for an example of the on-line help feature.

Third, the graphics and display functions are all contained in the **g.exe** program. When a plot is being displayed, pressing the "esc" key will return you to the main menu, or in those cases where there is no main menu, it will return you to the operating system. Pressing the "q" key exits the program. As an example, try typing **g view f001**, which will display the sequence stored in f001. There are two graphics functions that are slightly different: **g polezero** and **g apolezero**. These are completely menu driven and prompt you for all information.

Fourth, IIR filters (which are discussed later in this chapter and in Chapter 3) have the form $H(z) = B(z)/A(z)$. These files are stored with the numerator and denominator coefficients listed separately. To illustrate this, type **f003** to the screen and observe that the numerator and denominator coefficients are easily identifiable. More is said about the file structure for IIR filters in Exercise 1.4.4.

Fifth, to invoke any of the DSP functions, just type the function name. You will then be asked for any required arguments, such as the name of the input file, the name of the output file, and any appropriate function parameters. Alternatively, these can be included on the command line as shown below:

f function arg1 arg2 ....

The ordering of the arguments will vary from function to function, but normally the input file(s) are listed first, followed by the output file(s), and then any floating point or integer parameters that are required. The program will ask for any arguments that you omit. For example, consider the function **f add**, which has two inputs and one output. Assume that $x[n]$ is the signal stored in the file **f001**. The operation

$$y[n] = x[n] + x[n]$$

can be implemented by typing

f add f001 f001 yn

where **yn** is the output file containing $y[n]$. Try this and then display **yn** using **g view**. Remember that the *"esc"* or *"q"* keys will allow you to exit the function.

Finally, sequences can be complex valued. Complex numbers are stored as ordered pairs containing the real and imaginary parts. For example, the complex number $2.5+j5.3$ is represented by the pair 2.5 5.3. Notice that the real and imaginary parts are separated by a space. Manipulating complex sequences is similar to manipulating real ones. For example, the operation

$$y[n] = (2 + j2)x[n]$$

can be realized using the **f gain** function. Type

f gain f001 yn

where again we assume $x[n]$ is the signal contained in **f001**. You will be prompted for the value of the gain. Specify **2 2** corresponding to the complex number $2 + j2$. Alternatively, the value of the gain can be specified on the command line by typing

f gain vn yn 2 2

Try displaying **yn** using **g view**. Notice that now a menu appears requesting options regarding how you wish this complex sequence to be displayed.