

ARPACK

Users' Guide

SOFTWARE • ENVIRONMENTS • TOOLS

The series includes handbooks and software guides, as well as monographs on practical implementation of computational methods, environments, and tools.

The focus is on making recent developments available in a practical format to researchers and other users of these methods and tools.

Editor-in-Chief

Jack J. Dongarra

University of Tennessee and Oak Ridge National Laboratory

Editorial Board

James W. Demmel, *University of California, Berkeley*

Dennis Gannon, *Indiana University*

Eric Grosse, *AT&T Bell Laboratories*

Ken Kennedy, *Rice University*

Jorge J. Moré, *Argonne National Laboratory*

Software, Environments, and Tools

J. J. Dongarra, J. R. Bunch, C. B. Moler, and G. W. Stewart, *LINPACK Users' Guide*

Jack J. Dongarra, Iain S. Duff, Danny C. Sorensen, and Henk van der Vorst, *Solving Linear Systems on Vector and Shared Memory Computers*

E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen, *LAPACK Users' Guide, Second Edition*

Richard Barrett, Michael Berry, Tony F. Chan, James Demmel, June Donato, Jack Dongarra, Victor Eijkhout, Roldan Pozo, Charles Romine, and Henk van der Vorst, *Templates for the Solution of Linear Systems: Building Blocks for Iterative Methods*

Roger W. Hockney, *The Science of Computer Benchmarking*

Françoise Chaitin-Chatelin and Valérie Frayssé, *Lectures on Finite Precision Computations*

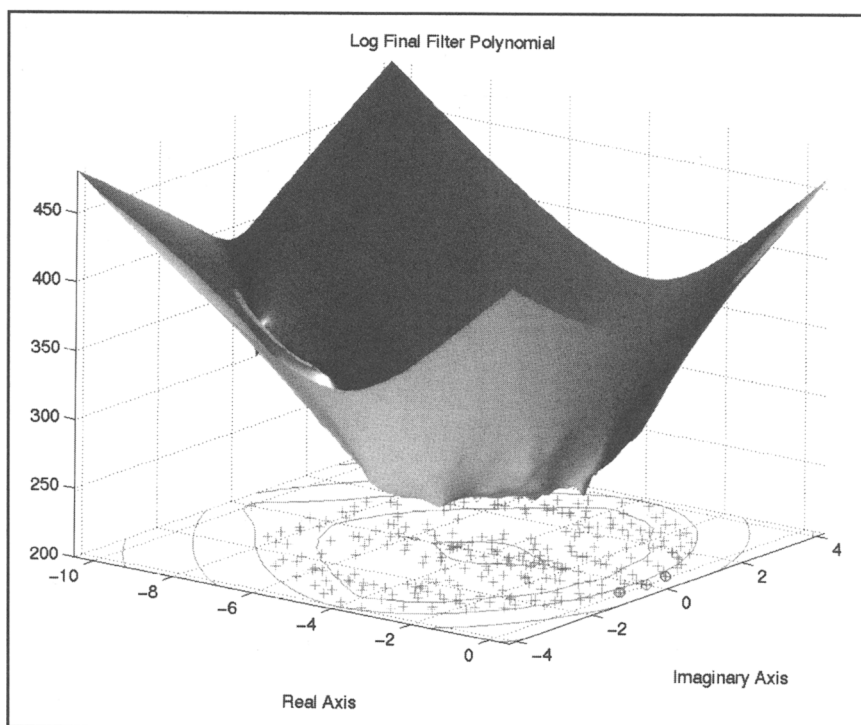
Greg Astfalk, editor, *Applications on Advanced Architecture Computers*

L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petit, K. Stanley, D. Walker, R. C. Whaley, *ScaLAPACK Users' Guide*

Randolph E. Bank, *PLTMG: A Software Package for Solving Elliptic Partial Differential Equations, Users' Guide 8.0*

R. B. Lehoucq, D. C. Sorensen, C. Yang, *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*

ARPACK



Users' Guide

Solution of Large-Scale Eigenvalue Problems
with Implicitly Restarted Arnoldi Methods

R. B. Lehoucq • D. C. Sorensen • C. Yang

Argonne National Laboratory
Argonne, Illinois

Rice University
Houston, Texas

Rice University
Houston, Texas

siam.


Society for Industrial and Applied Mathematics

Copyright ©1998 by the Society for Industrial and Applied Mathematics.

10 9 8 7 6 5 4 3 2 1

All rights reserved. Printed in the United States of America. No part of this book may be reproduced, stored, or transmitted in any manner without the written permission of the publisher. For information, write to the Society for Industrial and Applied Mathematics, 3600 University City Science Center, Philadelphia, PA 19104-2688.

No warranties, express or implied, are made by the publisher, authors, and their employers that the programs contained in this volume are free of error. They should not be relied on as the sole basis to solve a problem whose incorrect solutions could result in injury to person or property. If the programs are employed in such a manner, it is at the user's own risk and the publisher, authors, and their employers disclaim all liability for such misuse.

 The royalties from the sales of this book are being placed in a fund to help students attend SIAM meetings and other SIAM-related activities. This fund is administered by SIAM and qualified individuals are encouraged to write directly to SIAM for guidelines.

Library of Congress Catalog Card Number: 98-60204

ISBN 0-89871-407-9

siam is a registered trademark.

Contents

List of Figures	ix
List of Tables	xi
Preface	xiii
1 Introduction to ARPACK	1
1.1 Important Features	2
1.2 Getting Started	3
1.3 Reverse Communication Interface	3
1.4 Availability	3
1.5 Installation	4
1.6 Documentation	5
1.7 Dependence on LAPACK and BLAS	5
1.8 Expected Performance	6
1.9 P_ARPACK	6
1.10 Contributed Additions	6
1.11 Trouble Shooting and Problems	7
2 Getting Started with ARPACK	9
2.1 Directory Structure and Contents	9
2.2 Getting Started	10
2.3 An Example for a Symmetric Eigenvalue Problem	11
2.3.1 The Reverse Communication Interface	12
2.3.2 Postprocessing for Eigenvalues and Eigenvectors	14
2.3.3 Setting up the Problem	14
2.3.4 Storage Declarations	16
2.3.5 Stopping Criterion	16
2.3.6 Initial Parameter Settings	17
2.3.7 Setting the Starting Vector	18
2.3.8 Trace Debugging Capability	18
3 General Use of ARPACK	21
3.1 Naming Conventions, Precisions, and Types	21
3.2 Shift and Invert Spectral Transformation Mode	22

3.2.1	M is Hermitian Positive Definite	25
3.2.2	M is NOT Hermitian Positive Semidefinite	26
3.3	Reverse Communication for Shift-Invert	26
3.3.1	Shift and Invert on a Generalized Eigenproblem	29
3.4	Using the Computational Modes	29
3.5	Computational Modes for Real Symmetric Problems	31
3.6	Postprocessing for Eigenvectors Using dseupd	33
3.7	Computational Modes for Real Nonsymmetric Problems	34
3.8	Postprocessing for Eigenvectors Using dneupd	36
3.9	Computational Modes for Complex Problems	38
3.10	Postprocessing for Eigenvectors Using zneupd	40
4	The Implicitly Restarted Arnoldi Method	43
4.1	Structure of the Eigenvalue Problem	44
4.2	Krylov Subspaces and Projection Methods	47
4.3	The Arnoldi Factorization	49
4.4	Restarting the Arnoldi Method	52
4.4.1	Implicit Restarting	52
4.4.2	Block Methods	58
4.5	The Generalized Eigenvalue Problem	59
4.5.1	Structure of the Spectral Transformation	60
4.5.2	Eigenvector/Null-Space Purification	62
4.6	Stopping Criterion	64
5	Computational Routines	67
5.1	ARPACK Subroutines	69
5.1.1	XYaupd	69
5.1.2	XYaup2	69
5.1.3	XYaitr	71
5.1.4	Xgetv0	72
5.1.5	Xneigh	72
5.1.6	[s,d]seigt	72
5.1.7	[s,d]Yconv	73
5.1.8	XYapps	73
5.1.9	XYeupd	73
5.2	LAPACK routines used by ARPACK	75
5.3	BLAS routines used by ARPACK	75
A	Templates and Driver Routines	79
A.1	Symmetric Drivers	80
A.1.1	Selecting a Symmetric Driver	80
A.1.2	Identify OP and B for the Driver	84
A.1.3	The Reverse Communication Interface	84
A.1.4	Modify the Problem-Dependent Variables	88
A.1.5	Postprocessing and Accuracy Checking	90

A.2	Real Nonsymmetric Drivers	90
A.2.1	Selecting a Nonsymmetric Driver	91
A.2.2	Identify $\mathbf{0P}$ and \mathbf{B} for the Driver	93
A.2.3	The Reverse Communication Interface	94
A.2.4	Modify the Problem-Dependent Variables	97
A.2.5	Postprocessing and Accuracy Checking	99
A.3	Complex Drivers	99
A.3.1	Selecting a Complex Arithmetic Driver	100
A.3.2	Identify $\mathbf{0P}$ and \mathbf{B} for the Driver to be Modified	102
A.3.3	The Reverse Communication Interface	102
A.3.4	Modify the Problem-Dependent Variables	104
A.3.5	Postprocessing and Accuracy Checking	106
A.4	Band Drivers	106
A.4.1	Selecting a Band Storage Driver	108
A.4.2	Storing the Band Matrix Correctly	108
A.4.3	Modify Problem-Dependent Variables	109
A.4.4	Modify Other Variables if Necessary	109
A.4.5	Accuracy Checking	110
A.5	The Singular Value Decomposition	110
A.5.1	The SVD Drivers	112
B	Tracking the Progress of ARPACK	113
B.1	Obtaining Trace Output	113
B.2	Check-Pointing ARPACK	116
C	The XYaupd ARPACK Routines	121
C.1	DSAUPD	122
C.2	DNAUPD	125
C.3	ZNAUPD	128
	Bibliography	133
	Index	136

List of Figures

1.1	An example of the reverse communication interface used by ARPACK.	4
2.1	The ARPACK directory structure.	11
2.2	The reverse communication interface in <code>dssimp</code>	13
2.3	Postprocessing for eigenvalues and eigenvectors using <code>desupd</code> . .	15
2.4	Storage declarations needed for ARPACK subroutine <code>dsaupd</code> . .	17
2.5	How to initiate the trace debugging capability in ARPACK. . .	19
2.6	Output from a debug session for <code>dsaupd</code>	20
3.1	Reverse communication interface for shift-invert.	27
3.2	Reverse communication interface for shift-invert (contd). . . .	28
3.3	Calling the ARPACK subroutine <code>dnaupd</code>	30
3.4	Calling the ARPACK subroutine <code>dsaupd</code>	31
3.5	Postprocessing for eigenvectors using <code>dseupd</code>	33
3.6	Calling sequence of subroutine <code>dnaupd</code>	34
3.7	Postprocessing for eigenvectors using <code>dneupd</code>	37
3.8	Calling the ARPACK subroutine <code>znaupd</code>	39
3.9	Postprocessing for eigenvectors using <code>cneupd</code>	40
4.1	The implicitly restarted Arnoldi method in ARPACK.	44
4.2	Algorithm 1: Shifted QR-iteration.	47
4.3	Algorithm 2: The k -step Arnoldi factorization.	51
4.4	Algorithm 3: Implicitly restarted Arnoldi method (IRAM). . .	54
4.5	The set of rectangles represents the matrix equation $\mathbf{V}_m \mathbf{H}_m + \mathbf{f}_m \mathbf{e}_m^T$ of an Arnoldi factorization. The unshaded region on the right is a zero matrix of $m - 1$ columns.	55
4.6	After performing $m - k$ implicitly shifted QR steps on \mathbf{H}_m , the middle set of pictures illustrates $\mathbf{V}_m \mathbf{Q}_m \mathbf{H}_m^+ + \mathbf{f}_m \mathbf{e}_m^T \mathbf{Q}_m$. The last p columns of $\mathbf{f}_m \mathbf{e}_m^T \mathbf{Q}_m$ are nonzero because of the QR iteration.	55
4.7	An implicitly restarted length k Arnoldi factorization results after discarding the last $m - k$ columns.	55
4.8	Total filter polynomial from an IRAM iteration.	57
4.9	Total filter polynomial with spectral transformation.	61
5.1	<code>XYaupd</code> : Implementation of the IRAM/IRLM in ARPACK. . .	68

5.2	Outline of algorithm used by subroutine <code>XYeupd</code> to compute Schur vectors and possibly eigenvectors.	74
A.1	Reverse communication structure.	85
A.2	Compute $\mathbf{w} \leftarrow \mathbf{A}^T \mathbf{A} \mathbf{v}$ by blocks.	111
B.1	Sample output produced by <code>dsaupd</code>	114
B.2	The include file <code>debug.h</code>	116
B.3	Reading in a previous state with the example program <code>dssave</code>	117
B.4	Writing a state with the example program <code>dssave</code>	118
B.5	Writing a state with the example program <code>dssave</code> (contd).	119

List of Tables

2.1	List of the simple drivers illustrating the use of ARPACK. . . .	11
2.2	Parameters for the top-level ARPACK routines.	14
3.1	Available precisions and data types for ARPACK.	22
3.2	Double-precision top-level routines in subdirectory SRC.	23
3.3	The various settings for the argument <code>which</code> in <code>_saupd</code>	32
3.4	Possible settings for the argument <code>which</code> in <code>_naupd</code>	35
5.1	Description of the auxiliary subroutines of ARPACK.	70
5.2	Description of the LAPACK computational routines used by ARPACK.	76
5.3	Description of LAPACK auxiliary routines used by ARPACK.	76
5.4	Description of Level three BLAS used by ARPACK.	77
5.5	Description of Level two BLAS used by ARPACK.	77
5.6	Description of Level one BLAS used by ARPACK.	77
A.1	The functionality of the symmetric drivers.	81
A.2	The operators <code>OP</code> and <code>B</code> for <code>dsaupd</code>	84
A.3	The eigenvalues of interest for symmetric problems.	89
A.4	The functionality of the nonsymmetric drivers.	91
A.5	The operators <code>OP</code> and <code>B</code> for <code>dnaupd</code>	94
A.6	The eigenvalues of interest for nonsymmetric problems.	98
A.7	The functionality of the complex arithmetic drivers.	100
A.8	The operators <code>OP</code> and <code>B</code> for <code>znaupd</code>	102
A.9	The eigenvalues of interest for complex arithmetic problems.	105
A.10	Band storage drivers for symmetric problems.	107
A.11	Band storage drivers for nonsymmetric problems.	107
A.12	Band storage drivers for complex arithmetic problems.	108
B.1	Description of the message-level settings for ARPACK.	115

Preface

The development of ARPACK began as a research code written in Matlab¹ and then in Fortran 77 in 1990. Initially, the code was developed to study and verify the properties of the implicitly restarted Arnoldi method described in [44]. Preliminary experience with that code showed considerable promise in performance and also seemed to provide a solid foundation for the development of serious mathematical software for large structured eigenvalue problems.

During the academic year 1991–92, Dr. Phuong Vu (at that time with Cray Research) was granted permission to work in a half-time appointment to the NSF Center for Research on Parallel Computation at Rice University on the development of ARPACK. At the outset, we attempted to design the software to be efficient and portable on conventional high performance computing architectures. Of course, our design was also intended to be easily modified to effectively utilize a variety of parallel architectures (resulting in P_ARPACK). Phuong’s experience with users at Cray Research suggested that a reverse communication interface was essential. We are deeply indebted to Phuong for the design and implementation of this interface. He set the coding and documentation style and developed the initial implementation of all of the basic computational routines for real (single- and double-precision) matrices. His fundamental design has served us well as we have improved and expanded upon the package over the past few years.

We wish to thank the numerous users with applications and also our fellow numerical analysts who worked with initial “alpha” and then “beta” versions of the code. Their feedback and patience throughout this development has been invaluable. This interaction has often given us a wonderful sense of community and the words “it solved my problem!” always managed to brighten up the drudgery of developing and maintaining the software. In particular, we would like to mention Jean-Philippe Brunet, Daniela Calvetti, Lawrence Cowsar, Olivier Daube, David Day, Stewart Edwards, Ralph T. Goodwin III, Ed Hayes, Lennart Johnsson, Michiel Kooper, Karl Meerbergen, Frank Milde, Seymour Parter, Phil Pendergast, George Phillips, John Red-Horse, Lothar Reichel, Tod Romo, Will Sawyer, Jennifer Scott, Rajesh Kumar Singh, Allison Smith, Allister Spence, Zdenko Tomasic, Henk Van der Vorst.

¹Matlab is a registered trademark of The MathWorks, Inc., 24 Prime Park Way, Natick, MA 01760, USA, tel. 508-647-7000, fax 508-647-7001, info@mathworks.com, <http://www.mathworks.com>.

We wish to give special thanks to Kristi Maschhoff. She implemented the modifications required to produce the portable and parallel package P_ARPACK, redesigned both the makefiles and distribution mechanisms for ARPACK and P_ARPACK, and set up the ARPACK web site.

ARPACK is freely available through the world wide web and by anonymous ftp (see Chapter 1). It relies heavily upon the LAPACK software [1] and upon the BLAS [21, 11, 10]. Portability with performance, accuracy, and robustness is a direct consequence. We are greatly indebted to the authors of that software and more generally to the larger numerical analysis community that has contributed in many ways to its development.

Finally, we would like to thank the National Science Foundation, DARPA, and the Department of Energy for their generous support of this project (see below for full citation).

How to use this Guide

This users' guide is not intended to be read sequentially. There is a great deal of repetition amongst several of the subsections in Chapter 3 and also in Appendix A. We decided that it would be better to discuss each major problem class: real symmetric, real nonsymmetric, and complex as complete individual units even though this inevitably resulted in redundancy. We expect users to turn to the section that discusses the problem class of interest and to find everything related to that class in one complete section. We think this is preferable to searching back and forth to reference a general description in order to understand a special case of interest.

Chapter 1 gives an overview and contains general information. Chapter 2 provides installation instructions and describes how to get started. It is recommended for those who are just beginning with eigenvalue computations and also for those who are unfamiliar with reverse communication. Chapter 3 gives a detailed description of how to use all of the capabilities of ARPACK. Those wishing to learn a little about the underlying numerical methods should turn to Chapter 4. This discussion provides a broad overview of the methods. It gives a reasonably detailed description of the Arnoldi process with implicit restarting and what to expect. It also attempts to provide some understanding of the spectral transformation. Numerous references are provided for those who desire a more detailed level of understanding. Chapter 5 discusses implementation and usage details within the main computational routines. Experienced users of large-scale eigenvalue methods can probably turn directly to Appendix A and find the discussion of the driver routine that is appropriate for their problem. These drivers are intended to be used as templates that are easily modified for a particular application. Trace debugging and check-pointing are discussed in Appendix B. Finally, there are listings of the top-level reverse communication interface routines `XYaupd` for reference. The source code for all of the computational routines is available with the distribution. Each of these is

fully documented in the header and many users have found that documentation sufficient to get started with.

Research Funding of ARPACK

Financial support for this work was provided in part by the National Science Foundation cooperative agreement CCR-912008 and by the ARPA contract number DAAL03-91-C-0047 (administered by the U.S. Army Research Office). R. B. Lehoucq was also supported in part by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under contract W-31-109-Eng-38.