

ARPACK

Users' Guide

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 <code>dseupd</code>	33
3.7	Computational Modes for Real Nonsymmetric Problems	34
3.8	Postprocessing for Eigenvectors Using <code>dneupd</code>	36
3.9	Computational Modes for Complex Problems	38
3.10	Postprocessing for Eigenvectors Using <code>zneupd</code>	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	<code>XYaupd</code>	69
5.1.2	<code>XYaup2</code>	69
5.1.3	<code>XYaitr</code>	71
5.1.4	<code>Xgetv0</code>	72
5.1.5	<code>Xneigh</code>	72
5.1.6	<code>[s,d]seigt</code>	72
5.1.7	<code>[s,d]Yconv</code>	73
5.1.8	<code>XYapps</code>	73
5.1.9	<code>XYeupd</code>	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 <code>GP</code> and <code>B</code> 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 $\mathbf{QP}$ 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 $\mathbf{QP}$ 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>dasimp</code>	13
2.3	Postprocessing for eigenvalues and eigenvectors using <code>dasupd</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 $V_m H_m + f_m e_m^T$ 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 H_m , the middle set of pictures illustrates $V_m Q_m H_m^* + f_m e_m^T Q_m$. The last p columns of $f_m e_m^T Q_m$ 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