Algorithm 880: A Testing Infrastructure for Symmetric Tridiagonal Eigensolvers

OSNI A. MARQUES and CHRISTOF VÖMEL
Lawrence Berkeley National Laboratory
and
JAMES W. DEMMEL and BERESFORD N. PARLETT
University of California, Berkeley

LAPACK is often mentioned as a positive example of a software library that encapsulates complex, robust, and widely used numerical algorithms for a wide range of applications. At installation time, the user has the option of running a (limited) number of test cases to verify the integrity of the installation process. On the algorithm developer's side, however, more exhaustive tests are usually performed to study algorithm behavior on a variety of problem settings and also computer architectures. In this process, difficult test cases need to be found that reflect particular challenges of an application or push algorithms to extreme behavior. These tests are then assembled into a comprehensive collection, therefore making it possible for any new or competing algorithm to be stressed in a similar way. This article describes an infrastructure for exhaustively testing the symmetric tridiagonal eigensolvers implemented in LAPACK. It consists of two parts: a selection of carefully chosen test matrices with particular idiosyncrasies and a portable testing framework that allows for easy testing and data processing. The tester facilitates experiments with algorithmic choices, parameter and threshold studies, and performance comparisons on different architectures.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra

General Terms: Algorithms, Design

This work was partly supported by a grant from the National Science Foundation (Cooperative Agreement no. ACI-9619020), and by the Director, Office of Computational and Technology Research, Division of Mathematical, Information, and Computational Sciences of the U.S. Department of Energy under contract No. DE-AC03-76SF00098.

Authors' addresses: O. A. Marques, C. Vömel: Lawrence Berkeley National Laboratory, Computational Research Division, 1 Cyclotron Road, MS 50F-1650, Berkeley, CA 94720; email: {OAMarques,CVoemel}@lbl.gov. J. W. Demmel, B. N. Parlett: Mathematics Department and Computer Science Division, University of California, Berkeley, CA 94720; email: demmel@cs.berkeley.edu; parlett@math.berkeley.edu.

© 2008 ACM 0098-3500/2008/07-ART8 $5.00 DOI 10.1145/1377603.1377611 http://doi.acm.org/10.1145/1377603.1377611

Additional Key Words and Phrases: Eigenvalues, eigenvectors, symmetric matrix, LAPACK, accuracy, performance, test matrices, numerical software, design, implementation, testing

ACM Reference Format:

Marques, O. A., Vömel, C., Demmel, J. W., and Parlett, B. N. 2008. Algorithm 880: A testing infrastructure for symmetric tridagonal eigensolvers. ACM Trans. Math. Softw., 35, 1, Article 8 (July 2008) 13 pages DOI = 10.1145/1377603.1377611 http://doi.acm.org/10.1145/1377603.1377611

1. INTRODUCTION

The LAPACK software library [Anderson et al. 1999] provides a variety of drivers and computational routines for the sequential solution of Numerical Linear Algebra Problems. In preparation for a new release of LAPACK in 2007, we found it necessary to create a comprehensive testing environment to evaluate and compare the latest versions of LAPACK's symmetric eigensolvers. This paper describes our infrastructure, consisting of a test program and a set of test matrices.

Test matrices were originally designed to validate new algorithms. A Gregory and Karney [1969] state: 'what is needed is a collection of numerical examples with which to test each algorithm as soon as it is proposed.' Today, aspects other than pure validation have become equally important, in particular performance-related ones. There exist a number of sparse matrix collections that are most commonly used to evaluate the performance of sparse direct solvers [Duff et al. 1989, 1992, 1997; Davis 2007]. Furthermore, Matlab provides a test matrix collection based on Higham [1991]. Matrix Market contains non-Hermitian eigenvalue problems from Bai et al. [1997]. Test matrices from regularization with particular singular values and vectors are given in Hansen [1995]. Finally, a test matrix generator of large sparse matrices with given spectrum for the evaluation of iterative methods was recently presented in Lee and Stewart [2006].

Our own approach was guided by the following goals:

(1) to facilitate a critical examination of algorithm parameters,
(2) to allow easy post-processing of test results by tools such as Matlab or Excel,
(3) to create an easily portable infrastructure for various architectures, and
(4) to provide interesting and challenging test cases.

LAPACK provides test programs for a basic verification of its drivers and computational routines at installation time. This is useful for detecting problems caused by too aggressive compiler optimization options or the incorrect handling of IEEE-754 arithmetic features such as ∞ and NaN, or 'Not A Number' [ANSI/IEEE 1985; IEEE 754 2002; Overton 2001].

Our new testing infrastructure, `stetester`, is more comprehensive and useful for deeper studies of algorithms than the LAPACK tester. Following the approach adopted in the development of LAPACK, the tester is written in Fortran. It has four main applications:

(1) to experiment with and select algorithmic variants as in Marques et al. [2006],

Table I. LAPACK Codes for Computing Eigenpairs of a Symmetric Tridiagonal Matrix of Dimension n. Note that Inverse Iteration and the MRRR Algorithm also Allow the Computation of Eigenpair Subsets at Reduced Cost

Algorithm name	LAPACK subroutine	Subset Feature	Workspace Real/Integer
QR algorithm	`steqr`	N	$2n-2/0$
Divide & Conquer	`stedc`	N	$1+4n+n^2/3+5n$
Bisection/Inverse Iteration	`stebz/stein`	Y	$5n/5n$
MRRR algorithm	`stegr`	Y	$18n/10n$

(2) to experiment with parameter and threshold choices such as those to be set in LAPACK's `ilaenv`,
(3) to carry out large scale performance comparisons such as Marques et al. [2006]; Demmel et al. [2006], and
(4) to validate and benchmark new algorithms such as Matsekh [2005].

A brief summary of LAPACK's symmetric eigensolvers for which this tester has been developed is given in Table I, but see also Anderson et al. [1999]; Bai et al. [2001].[1]

References for the individual algorithms follow:

—QR algorithm [Demmel 1997; Golub and van Loan 1996; Parlett 1998],
—Divide & Conquer [Bunch et al. 1978; Cuppen 1981; Gu and Eisenstat 1994, 1995],
—Bisection/Inverse Iteration [Dhillon 1998; Ipsen 1997],
—MRRR algorithm [Dhillon 1997; Parlett and Dhillon 1997, 2000; Dhillon and Parlett 2004a, 2004b; Dhillon et al. 2006].

While it is possible to perform some of the tasks of this testing environment in Matlab, our tester does have some benefits:

—Being written in Fortran, it is portable across architectures at no cost.
—Low-level access to compiler options, IEEE-754 directives, for example, for changing the rounding mode, algorithm parameters, and detailed performance measures is possible.
—The overhead with respect to memory is negligible so that large matrices can be tested.

One particularly useful feature is our collection of *interesting* test matrices. We are interested in two types, ones that

—exhibit the strengths, weaknesses, or idiosyncrasies of a particular algorithm, and/or
—stem from an important application and thus are relevant to a larger group of users.

[1]The workspace that is reported for Divide & Conquer corresponds to the case COMPZ = 'I'. The workspace that is reported for Bisection/Inverse Iteration is for `stevx`, the driver that combines `stebz` and `stein`.

Table II. LAPACK-Style Test Matrices with a Given Eigenvalue Distribution. For Distributions 1-5, the Parameter k can be Chosen as ulp^{-1} Like in the LAPACK Tester but Other Choices are also Possible, see the Manual

Type	Description
1	$\lambda_1 = 1,\ \lambda_i = \frac{1}{k},\ i = 2, 3, \dots n$
2	$\lambda_i = 1,\ i = 1, 2, \dots n-1,\ \lambda_n = \frac{1}{k}$
3	$\lambda_i = k^{-(\frac{i-1}{n-1})},\ i = 1, 2, \dots n$
4	$\lambda_i = 1 - (\frac{i-1}{n-1})(1 - \frac{1}{k}),\ i = 1, 2, \dots n$
5	n random numbers in the range $(\frac{1}{k}, 1)$, their logarithms are uniformly distributed
6	n random numbers from a specified distribution
7	$\lambda_i = ulp \times i,\ i = 1, 2, \dots n-1,\ \lambda_n = 1$
8	$\lambda_1 = ulp,\ \lambda_i = 1 + \sqrt{ulp} \times i,\ i = 2, 3, \dots n-1, \lambda_n = 2$
9	$\lambda_1 = 1,\ \lambda_i = \lambda_{i-1} + 100 \times ulp, i = 2, \dots n$

Matrix *classes* or *families* can be of additional use as they allow scalability studies. As an example, matrices with clustered eigenvalues are considered to be difficult. Inverse iteration requires Gram-Schmidt orthogonalization to guarantee orthogonality among eigenvectors. Likewise, the MRRR algorithm struggles with very tight clusters of eigenvalues, see the examples in Dhillon et al. [2005]. As a second example, in order to understand the complexity of the Divide-and-Conquer algorithm, it is crucial to study the matrix-dependent deflation process (see for example Demmel [1997]).

This article makes two contributions:

(1) The testing infrastructure with the features mentioned above.
(2) A documentation of the reasons for including certain test matrices and how they are interesting.

The rest of this article is organized as follows. In Section 2, we describe the matrix types that are currently available in the tester and justify their relevance as important test cases. Section 3 describes the design of the testing infrastructure. Concluding remarks are given in Section 4.

2. TEST MATRICES

2.1 Random Matrices with Given Eigenvalue Distributions

Table II lists test matrix classes that are of "LAPACK-style" that is, distributions that are already used in LAPACK's tester [Demmel and McKenney 1989], fabricated distributions, and matrices that are used in Dhillon [1997]. Given the eigenvalues $D = \text{diag}(\lambda)$, the matrix $A = Q^T DQ$ is formed using an orthogonal matrix Q generated from random entries. Subsequently, A is reduced to tridiagonal form using LAPACK's `sytrd`.

Here, and in the following, the name ulp, or a relative unit-in-the-last-place is a number computed by the LAPACK subroutine `lamch` for precision. For example, on a Pentium M and double precision, the value is $2.2204460492503131E-16$.

Table III. Built-in Matrices with Distinguishing Performance-Relevant Features

Type	Description
0	zero matrix
1	identity matrix
2	(1,2,1) tridiagonal matrix
3	Wilkinson-type tridiagonal matrix
4	Clement-type tridiagonal matrix
5	Legendre-type tridiagonal matrix
6	Laguerre-type tridiagonal matrix
7	Hermite-type tridiagonal matrix

These test matrices are useful for the following reasons:

—The user can freely specify an eigenvalue distribution.
—The matrices do not need to be stored and can be generated on the fly.
—Matrices of arbitrary size can be generated.
—Eigenvalue clusters can be created easily.

In contrast to the tester in the LAPACK distribution, test parameters are directly accessible and can be changed easily without recompilation.

In addition to the distributions listed in Table II, the user can also read a list of eigenvalues from a file and then generate a tridiagonal as above using the matrix generator `latms` from the existing LAPACK testing infrastructure. See Section 3 for a description of this feature.

2.2 Matrices with Interesting Properties

Table III lists a number of matrix types included in our collection. Their performance-relevant properties are explained in the following Sections 2.2.1 and 2.2.2.

2.2.1 *Classical Test Matrices.* The (1-2-1) matrix is defined as

$$T = \text{tridiag}\begin{pmatrix} & 1 & & 1 & & & 1 & \\ 2 & & 2 & & \cdots & & & 2 \\ & 1 & & 1 & & & 1 & \end{pmatrix},$$

the Wilkinson matrix is

$$W^{+}_{2m+1} = \text{tridiag}\begin{pmatrix} & 1 & & 1 & & & & & & 1 & \\ m & & m-1 & & \cdots & 1 & 0 & 1 & \cdots & m-1 & & m \\ & 1 & & 1 & & & & 1 & & 1 & \end{pmatrix},$$

and the (symmetrized) Clement matrix is given by

$$T = \text{tridiag}\begin{pmatrix} & \sqrt{n} & & \sqrt{2(n-1)} & & \sqrt{(n-1)2} & & \sqrt{n} & \\ 0 & & 0 & & \cdots & & 0 & & 0 \\ & \sqrt{n} & & \sqrt{2(n-1)} & & \sqrt{(n-1)2} & & \sqrt{n} & \end{pmatrix},$$

that is, the off-diagonal entries are $\sqrt{i(n+1-i)}, i = 1, \ldots, n$.

These matrices have a number of interesting features that may favor one algorithm over another. We give a short summary here, additional information and analysis can be found in Demmel et al. [2006].

—The (1-2-1) matrix [Gregory and Karney 1969; Higham 1991] is the archetype of the symmetric tridiagonal Toeplitz matrix. Its eigenvalues and -vectors are known analytically; see Godunov et al. [1993] for a derivation. The eigenvalue clustering is not very strong (although clustering becomes tighter with growing dimension). Thus the matrix is relatively easy for MRRR to tackle. On the other hand, the top and bottom eigenvector entries that govern deflation in the Divide and Conquer algorithm do not decay quickly making this matrix class a difficult one for this algorithm.

—Wilkinson matrices [Wilkinson 1965; Gregory and Karney 1969; Higham 1991] are the opposite of (1-2-1) matrices in that they strongly favor Divide and Conquer over the MRRR algorithm. The top and bottom eigenvector entries decay very quickly below the deflation threshold. It can be verified that Divide and Conquer deflates all but a small number of eigenvalues (the number depends on the precision and the deflation threshold). On the other hand, since almost all eigenvalues of W^+_{2m+1} come in pairs which are well separated but increasingly close, the representation tree generated by the MRRR algorithm is very broad and the overhead for the tree generation is considerable. (For a definition of the representation tree and a discussion of its importance for the MRRR algorithm; see for example Dhillon et al. [2006].)

—The symmetrized version of the Clement (also called Kac) matrix [Clement 1959; Gregory and Karney 1969; Higham 1991] represents one interesting example of Golub-Kahan type. Its eigenvalues are the integers $\pm(n)$, $\pm(n-2), \ldots$. From the computational point of view, the matrix defines all its eigenvalues to high relative accuracy. As all eigenvalues are well spaced, matrices of this type are easy for inverse iteration and MRRR. On the other hand, deflation in Divide and Conquer is limited (especially when the matrix dimension is odd) because the top and bottom entries of the eigenvectors do not decay quickly.

In addition to the matrices listed in Table III, the user can also read a tridiagonal matrix from a file. See Section 3 for a description of this feature.

2.2.2 *Tridiagonals from Classical Orthogonal Polynomials.* There is a direct correspondence between tridiagonal matrices and certain families of orthogonal polynomials, see for example [Golub 1973; Gautschi 1994, 2002, 2004, 2005; Parlett 1998]. Our presentation uses the notation from Section 22.7 in Abramowitz and Stegun [1965].[2] For $i \geq 1$, the three-term recurrence

$$\frac{a_{4i}}{a_{3i}} f_{i-1} + \left(\frac{-a_{2i}}{a_{3i}} - \lambda \right) f_i + \frac{a_{1i}}{a_{3i}} f_{i+1} = 0 \qquad (1)$$

[2]In Abramowitz and Stegun [1965], the coefficients of classical three-term recurrences are reported in the form $a_{1i} f_{i+1}(\lambda) = (a_{2i} + a_{3i}\lambda) f_i(\lambda) - a_{4i} f_{i-1}(\lambda)$, where $a_{3i} \neq 0$. We use the equivalent (1) from which the tridiagonal can be directly derived.

defines a (nonsymmetric) tridiagonal matrix which is similar to a symmetric one provided that the coefficients satisfy $a_{4i+1}a_{1i} \geq 0$. Along these lines, the following symmetric tridiagonal matrices can be derived from Section 22.7 in Abramowitz and Stegun [1965].

The Legendre recurrence, (22.7.10) in Abramowitz and Stegun [1965], yields

$$T = \text{tridiag}\begin{pmatrix} & 2/\sqrt{3\cdot 5} & & 3/\sqrt{5\cdot 7} & & n/\sqrt{(2n-1)(2n+1)} & \\ 0 & & 0 & & \cdots & & 0 \\ & 2/\sqrt{3\cdot 5} & & 3/\sqrt{5\cdot 7} & & n/\sqrt{(2n-1)(2n+1)} & \end{pmatrix};$$

that is, the off-diagonal entries are $i/\sqrt{(2i-1)(2i+1)}, i = 2, \ldots, n$; see also Godunov et al. [1993].

Using the Laguerre recurrence, (22.7.12) in Abramowitz and Stegun [1965], with $\alpha = 0$ and off-diagonal entries chosen to be positive, one obtains

$$T = \text{tridiag}\begin{pmatrix} & 2 & & 3 & & n-1 & & n & \\ 3 & & 5 & & \cdots & & 2n-1 & & 2n+1 \\ & 2 & & 3 & & n-1 & & n & \end{pmatrix}.$$

Lastly, the Hermite recurrence, (22.7.14) in Abramowitz and Stegun [1965], gives

$$T = \text{tridiag}\begin{pmatrix} & \sqrt{1} & & \sqrt{2} & & \sqrt{n-2} & & \sqrt{n-1} & \\ 0 & & 0 & & \cdots & & 0 & & 0 \\ & \sqrt{1} & & \sqrt{2} & & \sqrt{n-2} & & \sqrt{n-1} & \end{pmatrix}.$$

(Note that the Chebyshev polynomials result in symmetric tridiagonal Toeplitz matrices that are affine translates of the (1-2-1) matrix from Section 2.2.)

All of these matrices have a fairly spread-out spectrum: compared to the spectral diameter, their eigenvalues are not very strongly clustered. Consequently, the MRRR algorithm can cope with them very efficiently. On the other hand, these matrix classes are challenging for Divide & Conquer which does no or little deflation on them compared to other matrix types.

2.3 Glued Matrices

Glued matrices, in particular glued Wilkinson matrices, emerged as important test matrices in particular for the MRRR algorithm, see Dhillon et al. [2005]. An analysis of their spectral properties is given in Parlett and Vömel [2007].

For symmetric tridiagonal matrices $T_1, \ldots, T_p$, define

$$T = \left[\begin{array}{c|c|c} T_1 & & \\ \hline & \ddots & \\ \hline & & T_p \end{array}\right] + \sum_{i=1}^{p-1} \gamma_i \left(x_i y_i^T + y_i x_i^T\right) , \tag{2}$$

where $x_i, y_i, i = 1, \ldots, p-1$ are columns of the identity corresponding to the interfaces between the diagonal blocks. The quantities γ_i are the glue factors.

The tester allows the user to generate general glued matrices by taking any combination of glue factors and matrices generated from the types listed in Tables II and III; see Sections 2.4 and 3 for details.

2.4 Tridiagonal Matrices from Applications

In the following, we describe those tridiagonal matrices in our tester that stem from applications.

—Examples from applications in computational quantum chemistry and electronic structure calculations. The tridiagonal matrices stem from solving a Schrödinger equation [Dhillon 1997; Dhillon et al. 1997] and were provided by George Fann using the NWChem computational chemistry package [Kendall et al. 2000; Apra et al. 2005]. Their dimensions range from 120 to 2053. These matrices have clustered eigenvalues that require a large number of reorthogonalizations when inverse iteration is used. This motivated the development of the MRRR algorithm which can cope well with matrices from this type of application.

—Examples from sparse matrix collections. These include matrices from the BCSSTRUC1 set in the Harwell-Boeing Collection [Duff et al. 1989, 1992, 1997] and matrices from the Alemdar, NASA, and Cannizzo sets in the University of Florida Sparse Matrix Collection [Davis 2007]. The matrices are related to the modeling of power system networks, a finite-difference model for the shallow wave equations for the Atlantic and Indian Oceans, and finite-element problems. The dimension of the corresponding tridiagonal matrices range from 48 to 8012. These matrices were chosen for their spectrum which typically consists of a part with eigenvalues varying almost continuously and another one with several isolated large clusters of eigenvalues of varying tightness. A large number of reorthogonalization steps is required within these clusters when inverse iteration is used.

For small matrices, the tridiagonal form of the sparse matrices was obtained with LAPACK's tridiagonal reduction routine `sytrd`. For the larger matrices, we generated tridiagonals by means of a simple Lanczos algorithm without reorthogonalization [Demmel 1997; Parlett 1998] and a starting vector filled with ones. When Lanczos is run without reorthogonalization, multiple copies of the eigenvalues keep reappearing so that the resulting tridiagonal will have very close eigenvalues. In this case, we ran an external implementation of Lanczos for kn steps, where $k = 1, \dots, 4$ and n is the dimension of the original sparse matrix, to obtain tridiagonals of dimension kn.

2.5 What Can Be Learned

Algorithm development and systematic testing using multiple platforms and various matrix classes is a challenging task. A systematic comparison of LAPACK's eigensolvers with respect to floating point operations, time, and accuracy can be found in [Demmel et al. 2006]. To give the reader a glimpse of the findings, we describe a few interesting issues.

Why can the Divide & Conquer algorithm on a (1-2-1) matrix of dimension 2048 be almost twice as fast as on one of dimensions 2047 or 2049? Why, in contrast, does the MRRR algorithm run equally fast for all three of them? (Answer: Substantially more deflation occurs in Divide & Conquer for n = 2048. On the other hand, the eigenvalue distributions of the three matrices are so close that there is no runtime difference in MRRR.)

What impact has the BLAS library on tridiagonal eigensolvers? (Answer: Except for Divide & Conquer, LAPACK's tridiagonal eigensolvers do not make use of higher level BLAS. However, between the self-compiled reference BLAS from Netlib and ATLAS BLAS, we noted timing differences of up to a factor of 10 for the Divide & Conquer algorithm.)

If the MRRR algorithm uses significantly fewer floating point operations than Divide & Conquer, why can it still be slower on some matrices? (Answer: While Divide & Conquer spends a majority of its time on matrix-matrix multiplication, MRRR relies heavily on Sturm counts which have an expensive division operation in each step.)

3. DESIGN OF THE TESTING INFRASTRUCTURE

This section describes in detail the testing infrastructure `stetester`.

Section 3.1 describes performance measures and the criteria for evaluating numerical results.

Input data for the tester is specified by means of key words or macros. These macros are groups of characters that uniquely define a specific subset of the input data, such as matrix types and dimensions, matrices to be read from files, etc. The tester includes a parser that interprets the data and the appropriate matrix types, parameters, and eigensolvers to be called. This is described in Section 3.2. After completion, the tester can return a large number of statistics regarding the performed tests, see Section 3.3.

3.1 Test Criteria

3.1.1 *Performance Measures: Time, Flops, Flips.* By default, the Fortran 95 function `cpu_time` is used for cross-architecture portability. To obtain more detailed performance-related information on floating point operations or instructions, PAPI [Browne et al. 2000; Dongarra et al. 2007] can be used.

In order to produce accurate timing results for matrices of smaller dimension, the same test case can be run multiple times and the average runtime is reported. The number of tests can be set such that the sum of all run times is significantly larger than the timer resolution.

3.1.2 *Numerical Accuracy.* For a tridiagonal matrix T with computed eigenvectors $Z = [z_1\, z_2 \cdots z_m]$ and eigenvalues $W = \operatorname{diag}(w_1\, w_2 \cdots w_m)$, $m \leq n$, `stetester` performs the following accuracy tests by using the LAPACK routine `lansy`:

$$\frac{\|I - ZZ^T\|}{n \times ulp}, \quad \text{if } m = n, \tag{3}$$

$$\frac{\|I - Z^T Z\|}{n \times ulp}, \quad \text{if } m < n, \tag{4}$$

to measure the orthogonality of the computed eigenvectors, and

$$\frac{\|T - ZWZ^T\|}{\|T\| \times n \times ulp}, \quad \text{if } m = n, \tag{5}$$

$$\frac{\|Z^T TZ - W\|}{\|T\| \times n \times ulp}, \quad \text{if } m < n, \tag{6}$$

which measures the accuracy of the computed eigenpairs.

For large matrices these tests can take a significant amount of time, so the user is given the option of choosing how many diagonals of the involved arrays are actually computed. Because eigenvalues are sorted from smallest to largest, faulty results in the orthogonality tests are likely to be caused by eigenvectors related to eigenvalues that are not far apart from each other. By taking say 10% or less of the diagonals we can save a significant amount of CPU time in large cases.

For testing the subset functionality in `stevx` and `stegr`, we pursue two approaches. For computing eigenvalues from a given smallest to a given largest index (parameter RANGE='I'), the code can pick random pairs of integers between 1 and n. For computing eigenvalues within an interval (parameter RANGE='V'), the code can select random interval bounds between ($\lambda_{\min}$ and $\lambda_{\max}$) (which are obtained with bisection, subroutine `stebz`).

3.2 Input Data and Key Words

The interface to `stetester` is a text parser. On input, key words can be specified in any order in the input file, either in lower or upper case, with the corresponding subset of data that they define. Data can be separated by blanks or commas, and the character % is interpreted as the beginning of a comment. Therefore, a line in the input file that begins with a % is simply ignored. More details are given in the manual.

3.3 Output Files

One of the important features of the testing infrastructure is its ability to print data for post-processing. The following output files can be generated optionally:

—File `stetester.out.T`. Tridiagonal matrix written as triplets j, d_j, e_j for each case (including the current seed of the random number generator if applicable).

—File `stetester.out.W`. Eigenvalues obtained by each tridiagonal eigensolver called.

—File `stetester.out.Z`. Eigenvectors obtained by each tridiagonal eigensolver called.

—File `stetester.out.log`. Timing and results of the tests for orthogonality (3) and residual norm (5) for each tridiagonal eigensolver called.

—File `stetester.out.m`. Tridiagonal matrix, eigenvalues and eigenvectors written in Matlab notation. The tridiagonal is represented by arrays `D_i` (diagonal entries) and `E_i` (off-diagonal entries), where `i` is the case number. The eigenvalues and eigenvectors are given in arrays `W_i_s` and `Z_i_s`, respectively, where i is the case number and s identifies the tridiagonal eigensolver according to the algorithm list as used for parameter CALLST, see the manual. For example, 1 stands for steqr with COMPZ='V', 2 for stevx with RANGE='A', and so forth. This is to distinguish between results computed with different algorithms.

As an illustration, a simple Matlab-style output file is given in the manual.

4. CONCLUSIONS AND AVAILABILITY

The main strength of the testing infrastructure described here is its ability to easily generate, read, and process a variety of test cases for symmetric tridiagonal eigensolvers. Together with our collection of test matrices, it allows users to perform exhaustive tests on a large range of computing platforms, with various compilers, revealing not only interesting numerical behavior but also important performance issues. It is available upon request.

In our experience, the presented infrastructure is an invaluable tool. As a collection of interesting matrices, it constitutes a benchmark for algorithm evaluation. As portable software infrastructure, it can help identify failures in algorithms, devise mechanisms to make algorithms more robust and efficient, and compare algorithms across computer architectures. We therefore anticipate it to be helpful for users who are interested in similar studies.

Lastly, our infrastructure could be further extended in different ways. One possibility is to provide software for the solution of tridiagonal inverse eigenvalue problems [Gragg and Harrod 1984]. A second possible direction is the modification and application of our infrastructure software model to the testing of other functionalities of LAPACK.

ACKNOWLEDGMENTS

We thank P. Arbenz and the three referees for their careful reading and many detailed comments on this article as well as for their suggestions on how to improve the installation process of the software.

REFERENCES

ABRAMOWITZ, M. AND STEGUN, I. A. 1965. *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables* 9th Ed. Dover, New York, NY.

ANDERSON, E., BAI, Z., BISCHOF, C., BLACKFORD, S., DEMMEL, J., DONGARRA, J., CROZ, J. D., GREENBAUM, A., HAMMARLING, S., MCKENNEY, A., AND SORENSEN, D. 1999. *LAPACK User's Guide* 3rd Ed. SIAM, Philadelphia, PA.

ANSI/IEEE 1985. IEEE Standard for Binary Floating Point Arithmetic, Std 754-1985 ed. ANSI/IEEE, New York, NY.

APRA, E. ET AL. 2005. NWChem, a computational chemistry package for parallel computers, version 4.7. Tech. rep., Pacific Northwest National Laboratory, Richland, WA.

BAI, Z., DAY, D., DEMMEL, J. W., AND DONGARRA, J. J. 1997. A test matrix collection for non-Hermitian eigenvalue problems. Tech. rep. UT-CS-97-355, University of Tennessee, Knoxville, TN.

BAI, Z., DEMMEL, J., DONGARRA, J., RUHE, A., AND VAN DER VORST, H. 2000. *Templates for the Solution of Algebraic Eigenvalue Problems—A practical guide*. SIAM, Philadelphia, PA.

BROWNE, S., DONGARRA, J., GARNER, N., HO, G., AND MUCCI, P. 2000. A portable programming interface for performance evaluation on modern processors. *Int. J. High Perf. Comput. Appl. 14,* 3, 189–204.

BUNCH, J., NIELSEN, P., AND SORENSEN, D. 1978. Rank-one modification of the symmetric eigenproblem. *Numer. Math. 31*, 31–48.

CLEMENT, P. A. 1959. A class of triple-diagonal matrices for test purposes. *SIAM Rev. 1*, 50–52.

CUPPEN, J. J. M. 1981. A divide and conquer method for the symmetric tridiagonal eigenproblem. *Numer. Math. 36*, 177–195.

DAVIS, T. A. 1994–2007. University of Florida Sparse Matrix Collection. *NA Digest 92*, 42; *NA Digest 96*, 28; *NA Digest 97*, 23.

DEMMEL, J. W. 1997. *Applied Numerical Linear Algebra*. SIAM, Philadelphia, PA.

DEMMEL, J. W., MARQUES, O. A., PARLETT, B. N., AND VÖMEL, C. 2006. Lapack working note 183: Performance and accuracy of LAPACK's symmetric tridiagonal eigensolvers. Tech. rep., University of California, Berkeley.

DEMMEL, J. W. AND MCKENNEY, A. 1989. A test matrix generation suite. Tech. rep. Computer Science Department, Courant Institute, New York, NY.

DHILLON, I. S. 1997. A new $O(n^2)$ algorithm for the symmetric tridiagonal eigenvalue/eigenvector problem. Ph.D. thesis, University of California, Berkeley, CA.

DHILLON, I. S. 1998. Current inverse iteration software can fail. *BIT 38:4*, 685–704.

DHILLON, I. S., FANN, G., AND PARLETT, B. N. 1997. Application of a new algorithm for the symmetric eigenproblem to computational quantum chemistry. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*. SIAM, Philadelphia, PA.

DHILLON, I. S. AND PARLETT, B. N. 2004a. Multiple representations to compute orthogonal eigenvectors of symmetric tridiagonal matrices. *Linear Algebra Appl. 387*, 1–28.

DHILLON, I. S. AND PARLETT, B. N. 2004b. Orthogonal eigenvectors and relative gaps. *SIAM J. Matrix Analy. Appl. 25,* 3, 858–899.

DHILLON, I. S., PARLETT, B. N., AND VÖMEL, C. 2005. Glued matrices and the MRRR algorithm. *SIAM J. Sci. Comput. 27,* 2, 496–510.

DHILLON, I. S., PARLETT, B. N., AND VÖMEL, C. 2006. The design and implementation of the MRRR algorithm. *ACM Trans. Math. Softw. 32,* 4, 533–560.

DONGARRA, J. J., MOORE, S., MUCCI, P., SEYMOUR, K., TERPSTRA, D., AND YOU, H. 2007. Performance Application Programming Interface (PAPI). `http://icl.cs.utk.edu/papi/`.

DUFF, I. S., GRIMES, R. G., AND LEWIS, J. G. 1989. Sparse matrix test problems. *ACM Trans. Math. Softw. 15*, 1–14.

DUFF, I. S., GRIMES, R. G., AND LEWIS, J. G. 1992. Users' guide for the Harwell-Boeing sparse matrix collection (Release I). Tech. rep. RAL-TR-92-086, Atlas Centre, Rutherford Appleton Laboratory.

DUFF, I. S., GRIMES, R. G., AND LEWIS, J. G. 1997. The Rutherford-Boeing sparse matrix collection. Tech. rep. RAL-TR-97-031, Atlas Centre, Rutherford Appleton Laboratory: Tech. rep. ISSTECH-97-017 from Boeing Information & Support Services; Report TR/PA/97/36 from CERFACS, Toulouse.

GAUTSCHI, W. 1994. Algorithm 726: ORTHPOL–a package of routines for generating orthogonal polynomials and Gauss-type quadrature rules. *ACM Trans. Math. Software 20,* 1, 21–62.

GAUTSCHI, W. 2002. The interplay between classical analysis and (numerical) linear algebra—a tribute to Gene H. Golub. *Electr. Trans. Num. Analy. 13*, 119–147.

GAUTSCHI, W. 2004. *Orthogonal Polynomials: Computation and Approximation*. Oxford University Press, Oxford, UK.

GAUTSCHI, W. 2005. Orthogonal polynomials (in Matlab). *J. Comput. Appl. Math. 178*, 215–234.

GODUNOV, S. K., ANTONOV, A. G., KIRILJUK, O. P., AND KOSTIN, V. I. 1993. *Guaranteed Accuracy in Numerical Linear Algebra*. Kluwer Academic, Dordrecht, The Netherlands.

GOLUB, G. H. 1973. Some modified matrix eigenvalue problems. *SIAM Rev. 15*, 318–334.

GOLUB, G. H. AND VAN LOAN, C. 1996. *Matrix Computations* 3rd Ed. The John Hopkins University Press, Baltimore, MD.

GRAGG, W. B. AND HARROD, W. J. 1984. The numerically stable reconstruction of Jacobi spectral data. *Numer. Math. 44*, 317–336.

GREGORY, R. AND KARNEY, D. 1969. *A Collection of Matrices for Testing Computational Algorithms*. Wiley, New York, NY.

GU, M. AND EISENSTAT, S. C. 1994. A stable and efficient algorithm for the rank-one modification of the symmetric eigenproblem. *SIAM J. Matrix Analy. Appl. 15,* 4, 1266–1276.

GU, M. AND EISENSTAT, S. C. 1995. A divide-and-conquer algorithm for the symmetric tridiagonal eigenproblem. *SIAM J. Matrix Analy. Appl. 16,* 1, 172–191.

HANSEN, P. C. 1995. Test matrices for regularization methods. *SIAM J. Sci. Comput. 16,* 2, 506–512.

HIGHAM, N. J. 1991. Algorithm 694: A collection of test matrices in MATLAB. *ACM Trans. Math. Softw. 17,* 3, 289–305.

IEEE 754 2002. IEEE Standard for Binary Floating Point Arithmetic Revision. `grouper.ieee.org/groups/754`.

IPSEN, I. C. F. 1997. Computing an eigenvector with inverse iteration. *SIAM Rev. 39,* 2, 254–291.

KENDALL, R. A., APRÀ, E., BEMHOLDT, D. E., ET AL. 2000. High Performance Computational Chemistry: An overview of NWChem a distributed parallel application. *Comput. Phys. Comm. 128*, 260–283.

LEE, C. R. AND STEWART, G. W. 2006. Eigentest: A test matrix generator for large-scale eigenproblems. UMIACS TR-2006-07, CMSC TR-4783, University of Maryland, College Park, MD.

MARQUES, O. A., RIEDY, E. J., AND VÖMEL, C. 2006. Benefits of IEEE-754 features in modern symmetric tridiagonal eigensolvers. *SIAM J. Sci. Comput. 28,* 5, 1613–1633.

MATSEKH, A. M. 2005. The Godunov-inverse iteration: A fast and accurate solution to the symmetric tridiagonal eigenvalue problem. *Appl. Numer. Math. 54,* 2, 208–221.

OVERTON, M. 2001. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM, Philadelphia, PA.

PARLETT, B. N. 1998. *The Symmetric Eigenvalue Problem*. SIAM, Philadelphia, PA.

PARLETT, B. N. AND DHILLON, I. S. 1997. Fernando's solution to Wilkinson's problem: An application of double factorization. *Linear Algebra Appl. 267*, 247–279.

PARLETT, B. N. AND DHILLON, I. S. 2000. Relatively robust representations of symmetric tridiagonals. *Linear Algebra Appl. 309,* 1–3, 121–151.

PARLETT, B. N. AND VÖMEL, C. 2007. The spectrum of a glued matrix. Tech. rep. University of California, Berkeley. Submitted.

WILKINSON, J. H. 1965. *The Algebraic Eigenvalue Problem*. Oxford University Press, Oxford, UK.

Received October 2006; revised September 2007; accepted November 2007