`irbleigs`: A MATLAB program for computing a few eigenpairs of a large sparse Hermitian matrix

J. Baglama
University of Rhode Island
D. Calvetti
Case Western Reserve University
and
L. Reichel
Kent State University

`irbleigs` is a MATLAB program for computing a few eigenvalues and associated eigenvectors of a sparse Hermitian matrix of large order n. The matrix is accessed only through the evaluation of matrix-vector products. Working space of only a few n-vectors is required. The program implements a restarted block-Lanczos method. Judicious choices of acceleration polynomials make it possible to compute approximations of a few of the largest eigenvalues, a few of the smallest eigenvalues, or a few eigenvalues in the vicinity of a user-specified point on the real axis. `irbleigs` also can be applied to certain large generalized eigenproblems as well as to the computation of a few nearby singular values and associated right and left singular vectors of a large general matrix.

Categories and Subject Descriptors: G.1.3 [**Numerical Analysis**]: Numerical Linear Algebra; G.4 [**Mathematical Software**]: Documentation

General Terms: Algorithms, Documentation

Additional Key Words and Phrases: block Lanczos method, eigenvalue computation, generalized eigenproblem, singular values, polynomial acceleration

1. INTRODUCTION

Let A be a large, possibly sparse, Hermitian matrix with real or complex elements. Let n denote the order of A. `irbleigs` is a MATLAB implementation of an implicitly restarted block-Lanczos algorithm for the computation of approximations of a few, say k, eigenvalues and associated eigenvectors of A, where we assume that $k \ll n$. The program is designed to determine approximations of the k largest eigenvalues, the k smallest eigenvalues, or k eigenvalues in the vicinity of a user-specified

This research was supported in part by a BSU New Faculty Research Grant and by NSF grants DMS-0107841 and DMS-0107858.

James Baglama, Department of Mathematics, University of Rhode Island, Kingston, RI 02881, jbaglama@math.uri.edu. Daniela Calvetti, Department of Mathematics, Case Western Reserve University, Cleveland, OH 44106, dxc57@po.cwru.edu. Lothar Reichel, Department of Mathematical Sciences, Kent State University, Kent, OH 44242, reichel@math.kent.edu.

point on the real axis.

`irbleigs` requires only moderate computer storage in addition to storage of the computed eigenvector approximations. The matrix A is accessed via a user-supplied function for the evaluation of matrix-vector products. In particular, A does not have to be stored. Moreover, matrices of the form $A - sI$, for $s \in \mathbf{R}$ and I the identity matrix, do not have to be stored or factored; `irbleigs` does not require that linear systems of equations with matrices of this form be solved. Storage of only a few n-vectors is required, in addition to the storage of the computed eigenvectors. This design of `irbleigs` makes it applicable to computing a few eigenpairs of large matrices on personal computers and workstations with not very large fast storage.

`irbleigs` implements an implicitly restarted block-Lanczos algorithm. This algorithm is presented in detail in [Baglama et al. 2003], and is a development of an implicitly restarted block-Lanczos algorithm described in [Baglama et al. 1998]. The latter algorithm, in turn, is an extension of the implicitly restarted Lanczos algorithms presented in [Calvetti et al. 1994; Sorensen 1992].

Implicitly restarted Arnoldi and Lanczos algorithms were first proposed by Sorensen [Sorensen 1992], who advocated the use of acceleration polynomials determined by so-called exact shifts. ARPACK [Lehoucq et al. 1998] implements the implicitly restarted Arnoldi and Lanczos algorithms based on these acceleration polynomials. The MATLAB function `eigs` is a MATLAB interface to ARPACK. ARPACK and `eigs` can be used to compute approximations of a few eigenpairs of general $n \times n$ non-Hermitian matrices. By specializing `irbleigs` to Hermitian matrices, and by basing the program on a restarted block-Lanczos method, we obtain a code that often determines the correct number of eigenvalues near a multiple eigenvalue (somewhat more often than `eigs`). Moreover, `irbleigs` performs better than `eigs` when storage limitations permit only a few steps of the Lanczos or block-Lanczos algorithms to be carried out between restarts; see [Baglama et al. 2003] for computed examples.

The choice of acceleration polynomial is important for the performance of implicitly restarted methods for eigenvalue computation. The specialization to Hermitian matrices allows a different choice of acceleration polynomials than in ARPACK. Numerical examples that illustrate the performance of `irbleigs` are presented in [Baglama et al. 2003]. Further computed examples and comparisons with several methods for the computation of a few eigenpairs of large Hermitian matrices are reported in [Baglama et al. 1996; Baglama et al. 1998; Calvetti et al. 1994].

Section 2 outlines the implicitly restarted block-Lanczos algorithm on which the code is based, Section 3 discusses user-specified parameters of `irbleigs`, and Section 4 describes a few sample uses of the program.

2. THE IMPLICITLY RESTARTED BLOCK-LANCZOS METHOD

`irbleigs` implements an implicitly restarted block-Lanczos method. Let $r \geq 1$ denote the block-size. Given an $n \times r$ matrix V_r with orthonormal columns, m steps of the block-Lanczos method yield the block-Lanczos decomposition

$$AV_{mr} = V_{mr}T_{mr} + F_rE_r^*, \tag{1}$$

where $V_{mr} \in \mathbf{C}^{n\times mr}$, $V_{mr}I_{mr\times r} = V_r$, $V_{mr}^*V_{mr} = I_{mr}$ and $F_r \in \mathbf{C}^{n\times r}$ satisfies $V_{mr}^*F_r = 0$. Here $I_{mr} \in \mathbf{R}^{mr\times mr}$ denotes the identity matrix, the matrix $I_{mr\times r} \in$

$\mathbf{R}^{mr\times r}$ consists of the first r columns of I_{mr}, and the matrix $E_r \in \mathbf{R}^{mr\times r}$ consists of the last r columns of I_{mr}. The superscript * denotes transposition and, when applicable, complex conjugation. Finally, T_{mr} is an $mr \times mr$ Hermitian block-tridiagonal matrix with upper triangular nonsingular subdiagonal $r \times r$ blocks. To secure numerical orthogonality of the columns of V_{mr}, they are reorthogonalized. We tacitly assume that the decomposition (1) with the specified properties exists. A discussion of how `irbleigs` handles situations when such a decomposition does not exist is provided in [Baglama 2000; Baglama et al. 2003].

Let $\{\theta, y\}$ be an eigenvalue-eigenvector pair of the matrix T_{mr} and define the vector $x := V_{mr}y$. Then θ and x are commonly referred to as a Ritz value and a Ritz vector of A, respectively. It follows from (1) that the residual error $Ax - x\theta$ associated with the Ritz pair $\{\theta, x\}$ satisfies

$$\|Ax - x\theta\| = \|(AV_{mr} - V_{mr}T_{mr})y\| = \|F_r E_r^* y\|, \tag{2}$$

where $\|\cdot\|$ denotes the Euclidean vector norm or the associated induced matrix norm. Thus, the norm of the residual error can be computed without explicitly computing the Ritz vector x by evaluating the right-hand side of (2). When

$$\|F_r E_r^* y\| \leq \epsilon\eta(A), \tag{3}$$

for a user-specified value of ϵ, the Ritz value θ is accepted as an approximate eigenvalue of A. Here $\eta(A)$ denotes an inexpensively computable approximation of $\|A\|$. In `irbleigs`, $\eta(A)$ is the absolute value of the Ritz value of largest magnitude determined thus far during the computation.

Assume that the block-Lanczos decomposition (1) has been computed by m steps of the block-Lanczos algorithm, and let m be the largest number of block-Lanczos steps that we wish to carry out between restarts. Let the norm of the residual error (2) be larger than a specified tolerance for the Ritz pairs of interest. We then apply recursion formulas derived in [Baglama et al. 1998] to compute the matrix

$$U_r := p_m(A)V_r, \tag{4}$$

where p_m is a polynomial of degree m, to be specified below. We refer to p_m as an acceleration polynomial. Given the block-Lanczos decomposition (1), the matrix U_r can be computed without the evaluation of any additional matrix-vector products with A; see [Baglama et al. 1998] for details. Orthogonalization of the columns of U_r yields the matrix V_r^+; thus,

$$U_r = V_r^+ R_r^+, \qquad V_r^+ \in \mathbf{C}^{n\times r}, \qquad R_r^+ \in \mathbf{C}^{r\times r},$$

where $(V_r^+)^* V_r^+ = I_r$ and R_r^+ is upper triangular. The computations that determined the matrix V_r^+ from V_r are now repeated with the matrix V_r^+ replacing V_r. Thus, application of m steps of the block-Lanczos method to A with initial block V_r^+ yields the block-Lanczos decomposition

$$AV_{mr}^+ = V_{mr}^+ T_{mr}^+ + F_r^+ E_r^*.$$

If the desired Ritz values have not been determined with sufficient accuracy by this decomposition, then a new acceleration polynomial p_m^+ of degree m is chosen and the matrix

$$U_r^+ := p_m^+(A)V_r^+ \tag{5}$$

is evaluated in the same way as the matrix U_r in (4) without additional matrix-vector product evaluations with the matrix A. Combining (4) and (5) yields

$$U_r^+ = p_m^+(A)p_m(A)V_r(R_r^+)^{-1}.$$

Orthogonalization of the columns of U_r^+ now gives the matrix V_r^{++}. New block-Lanczos decompositions are evaluated in this manner until approximations of all desired eigenvalues and eigenvectors have been computed with sufficient accuracy.

The performance of the IRBL method crucially depends on the choice of the sequence of acceleration polynomials $p_m, p_m^+, \ldots$. We determine these polynomials by specifying their zeros so that the product of the computed acceleration polynomials is of large magnitude in a vicinity of the eigenvalues that we wish to compute and of small magnitude on the remaining part of the spectrum of A.

The zeros of the acceleration polynomials $p_m, p_m^+, \ldots$ are allocated on sets $\mathbf{K}$ that contain some of the undesired eigenvalues of A and none of the desired ones. For instance, if we wish to compute a few of the largest eigenvalues of A, then the sets $\mathbf{K}$ are intervals on the real axis to the left of the desired eigenvalues. When we wish to determine a few non-extreme eigenvalues, the sets $\mathbf{K}$ generally consist of two real intervals, one on each side of the set of desired eigenvalues. How the sets $\mathbf{K}$ are determined from the block-tridiagonal matrices T_{mr}, and from closely related matrices, evaluated during the computations is described in [Baglama et al. 2003]. The zeros are distributed like Leja points for the sets $\mathbf{K}$, and we refer to the zeros as weighted fast Leja points or as mapped fast Leja points, depending on how they are allocated; see [Baglama et al. 2003] for details. The arithmetic work to determine the zeros is typically negligible.

We remark that in addition to computing a few eigenvalues and eigenvectors of a Hermitian matrix A, `irbleigs` can also be used to determine a few eigenvalues and eigenvectors of generalized eigenvalue problems of the form

$$Ax = \lambda Mx, \tag{6}$$

where A is a Hermitian matrix and M is a Hermitian positive definite matrix, such that its Cholesky factorization can be computed. We denote the upper triangular Cholesky factor of M by R; thus $M = R^*R$.

Finally, we note that `irbleigs` can compute a few singular values and associated right and left singular vectors of a general $n_1 \times n_2$ matrix C by determining a few eigenvalues and associated eigenvectors of the Hermitian matrix

$$A = \begin{bmatrix} 0 & C \\ C^* & 0 \end{bmatrix} \in \mathbf{C}^{(n_1+n_2)\times(n_1+n_2)}; \tag{7}$$

see [Baglama et al. 2003] for a computed example. We remark that the `svds` function in MATLAB also can be applied to the computation of a few singular values and singular vectors of a general matrix. However, `svds` uses a shift-and-invert approach to determine non-extreme eigenvalues of the matrix (7), and therefore is not well suited for solving very large problems.

3. PARAMETERS

`irbleigs` allows a user to set parameters that specify the approximate location of the desired eigenvalues, determine the computer storage required and the choice

of acceleration polynomial, and affect the output produced. This section contains a brief description of these parameters. We begin with the input parameters in alphabetical order, followed by the output parameters. All parameters are assumed to take on numerical values, unless otherwise specified. The default value of each parameter is provided. The names of parameters are in *italics*. For instance, n denotes the parameter, whose value is the order n of the matrix A.

blsz Block-size of the block-Lanczos method. The parameter specifies the value of r in (1). Default value: $blsz = 3$.

cholM Indicates whether the upper triangular Cholesky factor R of the matrix M in the generalized eigenvalue problem (6) is available. If it is, then let $cholM = 1$ and replace the input matrix M by R, otherwise let $cholM = 0$. Default value: $cholM = 0$.

dispr When $dispr > 0$, the k desired Ritz values along with the corresponding residual errors (2) are displayed each iteration; $dispr = 0$ inhibits display of these quantities. Default value: $dispr = 0$.

eigvec A matrix of converged eigenvectors. When the matrix *eigvec* is nonempty, the computed eigenvectors are forced to be orthogonal to the columns of the matrix *eigvec*. No internal checking is performed to determine if the columns of *eigvec* are mutually orthogonal or if they are eigenvectors of the input matrix. Default value: *eigvec* = [ ].

endpt A three-letter string specifying the strategy for selecting the endpoint(s) closest to the desired eigenvalues. Admissible values for *endpt* are:

'FLT' - endpoint(s) closest to the desired eigenvalues are selected at each iteration independently of its (their) previous value(s). The endpoints are allowed to "float."

'MON' - all endpoints vary monotonically, so that the dampening intervals that make up the sets **K** are nested.

More details on how the parameter *endpt* affects the choice of the sets **K** can be found in [Baglama et al. 2003]. Default values: *endpt* = 'FLT' if the parameter *sigma* has a numerical value, and *endpt* = 'MON' if *sigma* = 'SE' or *sigma* = 'LE'.

funpar If an M-file is provided for the evaluation of matrix-vector products with the matrix A, then additional parameters that may be required to compute the matrix-vector products can be provided by the parameter *funpar*. This parameter can be numerical, a character, or a MATLAB structure. The parameters for the M-file for the evaluation of matrix-vector products must be given in the order ($X, n, blsz, funpar$). The M-file, called Afunc.m below, determines the output $Y = AX$. Default value: *funpar* = [ ].

k Number of desired eigenvalues. Default value: $k = 3$.

maxit Maximum number of restarts of the block-Lanczos method. Default value: $maxit = 100$.

maxdpol Maximum degree of the acceleration polynomial. Default values: $maxdpol = n$ if *sigma* has a numerical value, and $maxdpol = 200$ if *sigma* = 'SE' or *sigma* = 'LE'.

nbls Number of steps of the block-Lanczos method between restarts at the beginning of the computations. The parameter specifies the value of m in (1). The value may be adjusted during the computations, so that better sets $\mathbf{K}$, on which zeros of the acceleration polynomials are allocated, can be determined. The value of *nbls* may also be changed to reduce the storage requirement of the program; see [Baglama et al. 2003] for a discussion of the latter. Default value: $nbls = 3$. A larger value of *nbls* may, but is not guaranteed to, reduce the number of matrix-vector product evaluations required to satisfy the termination criterion.

permM Permutation vector for the rows and columns of the Hermitian positive definite matrix M in the generalized eigenvalue problem (6). The Cholesky factor R of the matrix $M(permM, permM)$ is computed, i.e., $M(permM, permM) = R^*R$. Permutation of the rows and columns of M may reduce the fill-in produced during the Cholesky factorization. For instance, the vector *permM* can be determined by the MATLAB function `symamd`. Default value: $permM = 1{:}n$, which gives no permutation.

sigma A two letter string or a numerical value specifying the location of the desired eigenvalues. Admissible values for the string are:

'LE' - `irbleigs` seeks to compute the k largest eigenvalues,

'SE' - `irbleigs` seeks to compute the k smallest eigenvalues.

When *sigma* has a numerical value, `irbleigs` seeks to compute k eigenvalues in a vicinity of this value. Default value: *sigma* = 'LE'.

sizint Size of the intervals that contain zeros of the acceleration polynomials. The intervals are smallest when $sizint = 1$ and largest when $sizint = (nbls - 1)blsz - k$. The length of the intervals grows with the integer-valued parameter $sizint \in [1, (nbls - 1)blsz - k]$. The exact sizes of the intervals depends on the distribution of computed Ritz and harmonic Ritz values of A. More details on how the parameter *sizint* affects the length of the intervals can be found in [Baglama et al. 2003]. Default value: $sizint = 1$.

tol Tolerance used for convergence check. The value of *tol* specifies the value of ϵ in (3). Default value: $tol = 1 \cdot 10^{-6}$.

v0 A matrix of *blsz* orthonormal starting vectors. By default, the *blsz* columns of *v0* are determined by orthogonalizing the columns generated by the MATLAB function `randn`$(n, blsz)$. This function determines an $n \times blsz$ matrix whose entries are random numbers from the standard normal distribution.

zertyp A two-letter string that specifies how the zeros of the acceleration polynomial are selected. Admissible values for the string are:

'WL' - weighted fast Leja points,

'ML' - mapped fast Leja points. Fast Leja points are computed for the interval $[-2, 2]$ and mapped to the interval $\mathbf{K}$. This option is not available when *sigma* has a numerical value.

Default values: 'ML' if *sigma* = 'SE' or *sigma* = 'LE', and 'WL' if *sigma* has a numerical value.

The parameter values for `irbleigs` can be supplied via a MATLAB structure with the field values as parameter names. This allows a user to supply and change parameter values easily; see Section 4 for examples. The input order is

$$(A, M, OPTS) \qquad \text{or} \qquad (\text{'Afunc'}, n, M, OPTS), \tag{8}$$

where the first input argument is either an $n \times n$ Hermitian matrix, here denoted by A, or the name of an M-file, here called Afunc.m, for the evaluation of matrix-vector products with the matrix A. Note that if the first argument is the name of an M-file, then the second argument must be the order n of the matrix.

The parameter denoted by M in (8) is optional; if it is omitted then eigenpairs of the matrix A are computed. If the parameter M is included and the input parameter $cholM = 0$, then eigenpairs of the generalized eigenvalue problem (6) are determined, where M is a Hermitian positive definite matrix. In the latter case, `irbleigs` computes the Cholesky factorization of M by calling the MATLAB function `chol`; see also the discussion of the parameter *permM*. Finally, if the parameter M is included and $cholM > 0$, then M is assumed be the Cholesky factor R of the Hermitian positive definite matrix in the right-hand side of (6).

The input argument $OPTS$ in (8) is optional. If supplied, it is a MATLAB structure that contains some or all of the input parameters in arbitrary order; see Section 4 for illustrations.

The M-file Afunc.m for evaluating the matrix-vector products with the matrix A can be written in MATLAB, C or FORTRAN. In the latter cases, it is interfaced with MATLAB using MEX files; see [MathWorks 1998]. The M-file Afunc.m must take an $n \times blsz$ matrix X as the first input argument, the integer n as the second input argument, *blsz* as the third input argument and *funpar* as an optional fourth argument; see the discussion related to the parameter *funpar*. Afunc.m must return the product AX.

There are four available output options:

`irbleigs`$(A, M, OPTS)$: Displays the desired eigenvalues.

$e =$ `irbleigs`$(A, M, OPTS)$: Returns the desired eigenvalues in the vector e.

$[V, D] =$ `irbleigs`$(A, M, OPTS)$: Returns the matrices V and D, where D is a diagonal matrix that contains the desired eigenvalues along the diagonal and the columns of the matrix V contain the corresponding orthonormal eigenvectors, such that $AV = VD$.

$[V, D, EXITINFO] =$ `irbleigs`$(A, M, OPTS)$: Returns the matrices V and D described above, as well as the array $EXITINFO$ with two entries. $EXITINFO(1) = 0$ signals normal exit from `irbleigs`; k eigenpairs have been computed with desired accuracy. The number of matrix-vector product evaluations with the matrix A is reported in $EXITINFO(2)$. One evaluation of the product AX, where X is an $n \times blsz$ matrix, counts as *blsz* matrix-vector product evaluations. If $EXITINFO(1) = 1$, then the maximum number of iterations, specified by the parameter *maxit*, have been carried out, but the k desired eigenpairs have not been computed with specified accuracy. In this situation, the matrices V and D contain Ritz pairs that

approximate wanted eigenpairs, but not all of them have been determined with specified accuracy. We remark that the parameter *nbls* might increase or decrease during the iterations and therefore the number of matrix-vector product evaluations with the matrix A might not be equal to the number of iterations times the input parameters *nbls* and *blsz*. Thus, if the program exits with $EXITINFO(1) = 1$, then the number of matrix-vector product evaluations reported in $EXITINFO(2)$ might not equal $maxit \times nbls \times blsz$ because *nbls* might have changed during the iterations.

The first three output options listed above return empty arrays if the maximum number of iterations, specified by the parameter *maxit*, is reached before k eigenpairs have been determined with desired accuracy. Only the last output option, which includes the array *EXITINFO*, yields Ritz pairs when the maximum number of iterations is reached before all wanted Ritz pairs have been computed with desired accuracy. These Ritz pairs can be used in a subsequent run of `irbleigs` to improve their accuracy; see Section 4 for an illustration.

4. SAMPLE EXPERIMENTS

In this section we present a few calling sequences for `irbleigs` and the corresponding output. All experiments were carried out using MATLAB version 6.0, with patches from The MathWorks to remedy the memory leakage, on a Gateway E-5200 workstation with two 450 MHz (512k cache) Pentium III processors and 128 MB (100 MHz) of memory. In all experiments reported below, A is a 100×100 matrix obtained by discretizing the 2-dimensional negative Laplace operator on the unit square by the standard 5-point stencil with Dirichlet boundary conditions. This matrix can be generated in MATLAB with the command

```
>> A = delsq(numgrid('S',12));
```

Note that MATLAB stores the matrix A in sparse format. The command

```
>> irbleigs(A)
```

makes `irbleigs` compute the 3 largest eigenvalues of A using the default values for all parameters. The output is

```
eigenvalues =
    7.8380
    7.6015
    7.6016
```

The parameter values can be changed by using the structure OPTS, whose field values are the parameter names. For example, to compute the smallest eigenvalues of A, the value of *sigma* should be set to 'SE'. This can be done by the command

```
>> OPTS = struct('sigma','SE');
```

or by setting

```
>> OPTS.sigma = 'SE';
```

Subsequently, the command

```
>> irbleigs(A,OPTS)
```

yields the 3 smallest eigenvalues of A:

```
eigenvalues =
    0.1620
    0.3985
    0.3985
```

Other parameters may be changed similarly. For instance, to determine 4 eigenvalues near 2 using a block-size of 2 with, in general, 8 consecutive steps by the block-Lanczos method between restarts, the following sequence of commands can be used:

```
>> OPTS.sigma = 2;
>> OPTS.k = 4;
>> OPTS.blsz = 2;
>> OPTS.nbls = 8;
```

Then the command

```
>> irbleigs(A,OPTS)
```

yields

```
eigenvalues =
    1.8594
    1.8594
    2.0329
    2.0329
```

If both eigenvalues and eigenvectors are desired, then `irbleigs` should be called as follows:

```
>> [V,D] = irbleigs(A,OPTS);
```

The computed eigenvalues make up the diagonal entries of the diagonal matrix D and the corresponding computed orthonormal eigenvectors are the columns of the matrix V.

Assume for the moment that an M-file called Afunc.m is available that takes as input $(X, 100, blsz)$, where X is a $100 \times blsz$ matrix, and yields the matrix $Y = AX$ as output. The command

```
>> [V,D] = irbleigs('Afunc',100,OPTS);
```

yields the eigenpairs of A specified by the structure OPTS.

In order to compute a few eigenpairs of a generalized eigenvalue problem (6) a user must provide the matrices A and M. The Hermitian matrix A can be supplied as described above. The matrix M must be Hermitian and positive definite. Either the matrix M or its Cholesky factor are supplied in numerical format, with the parameter *cholM* chosen accordingly. For instance, the commands

```
>> M = diag(1:100);
>> irbleigs(A,M)
```

yield the output

```
eigenvalues =
     4.2525
     2.0466
     1.3617
```

`irbleigs` returns the 3 largest eigenvalues of the generalized eigenvalue problem using default parameter values. If we instead would like to compute 5 eigenvalues near 0.2, then we issue the commands

```
>> OPTS.sigma = 0.2;
>> OPTS.k = 5;
>> irbleigs(A,M,OPTS)
```

and obtain the output

```
eigenvalues =
     []
```

The missing output indicates that `irbleigs` did not find the desired eigenvalues of the generalized eigenvalue problem with the specified values of the parameters k and *sigma*, and the default values of the remaining input parameters. Changing the output option to $[V, D, EXITINFO]$ allows `irbleigs` to output approximate Ritz pairs in the matrices D and V. Thus,

```
>> [V,D,EXITINFO] = irbleigs(A,M,OPTS);
>> EXITINFO
```

yields

```
EXITINFO =
     1  1191
```

The value 1 of $EXITINFO(1)$ shows that the maximum number of iterations have been carried out, and the value 1191 of $EXITINFO(2)$ displays the number of matrix-vector product evaluations with the matrix A that have been carried out before exit. Note that $EXITINFO(2)$ is larger than $maxit \times nbls \times blsz = 900$, because in order to determine a suitable set $\mathbf{K}$, `irbleigs` increased the value of *nbls* from 3 to 4 at the 4th restart of the block-Lanczos process.

The matrices V and D contain approximate eigenpairs. Increasing the value of *blsz* to 5 and letting the parameter *v0* be equal to the matrix V,

```
>> OPTS.blsz = 5;
>> OPTS.v0 = V;
>> irbleigs(A,M,OPTS)
```

yields the desired output

```
eigenvalues =
    0.1758
    0.1803
    0.1882
    0.2065
    0.2099
```

Note that *blsz* must be increased to 5, otherwise an error message will be printed saying that the starting matrix is of the wrong size, followed by program exit. An alternate approach to determining the desired eigenpairs is to increase the values of the parameters *nbls* and *maxit*.

Acknowledgement. We would like to thank Richard Lehoucq, Axel Ruhe and Mike Saunders for valuable comments.

REFERENCES

BAGLAMA, J. 2000. Dealing with linear dependence during the iterations of the restarted block Lanczos methods. *Numer. Algorithms 25*, 23–36.

BAGLAMA, J., CALVETTI, D., AND REICHEL, L. 1996. Iterative methods for the computation of a few eigenvalues of a large symmetric matrix. *BIT 36*, 400–421.

BAGLAMA, J., CALVETTI, D., AND REICHEL, L. 2003. IRBL: An implicitly restarted block Lanczos method for large-scale Hermitian eigenproblems. *SIAM J. Sci. Comput. 24*, 1650–1677.

BAGLAMA, J., CALVETTI, D., REICHEL, L., AND RUTTAN, A. 1998. Computation of a few small eigenvalues of a large matrix with applications to liquid crystal modeling. *J. Comput. Phys. 146*, 203–226.

CALVETTI, D., REICHEL, L., AND SORENSEN, D. C. 1994. An implicitly restarted Lanczos method for large symmetric eigenvalue problems. *Elec. Trans. Numer. Anal. 2*, 1–21.

LEHOUCQ, R. B., SORENSEN, D. C., AND YANG, C. 1998. *ARPACK Users' Guide: Solution of Large-Scale Eigenvalue Problems with Implicitly Restarted Arnoldi Methods*. SIAM, Philadelphia. Code available at web site http://www.caam.rice.edu/software/ARPACK.

MATHWORKS. 1998. *MATLAB Application Program Interface Guide, Version 5.*

SORENSEN, D. C. 1992. Implicit application of polynomial filters in a k-step Arnoldi method. *SIAM J. Matrix Anal. Appl. 13*, 357–385.