# PRIMER FOR THE MATLAB FUNCTIONS `IRLBA` AND `IRLBABLK`

JAMES BAGLAMA* AND LOTHAR REICHEL†

This guide describes implementation and use of the MATLAB functions `irlba` and `irlbablk`. The numerical methods on which these functions are based are presented in [1, 2], where further properties and details of the performance of `irlba` and `irlbablk` can be found.

**1. Files and installation.** The M-files `irlba.m` and `irlbablk.m` for the MATLAB functions `irlba` and `irlbablk` are available from Netlib at

*http://www.netlib.org/numeralgo/na26*

The codes have been developed and tested using MATLAB versions 6.5-7.2 by MathWorks. No other MathWorks products or toolboxes are required.

The installation process is described in Table 1.1. The MATLAB functions and data obtained with the installation are listed in Table 1.2.

TABLE 1.1
*Installation.*

| | Windows system | Unix system |
|---|---|---|
| Step 1. | Download the gzipped file na26.tgz from Netlib. | |
| Step 2 | Unzip the file na26.tgz. | |
| | Use, for example, Winzip to extract all files and store them in the directory IRLBA. | The command *tar xvfz na26.tgz* creates the directory IRLBA. Remove the archive with the command *rm na26.tgz*. |
| Step 3. | Start MATLAB. | |
| Step 4. | Add a path to the directory IRLBA of Step 2. | |

TABLE 1.2
*The files in na26.tgz.*

| File | Type | Size | Description |
|---|---|---|---|
| `Afunc.m` | MATLAB M-file | 1 KB | Sample MATLAB function for the evaluation of matrix-vector products to be used in `irlbablk.m`; see Table 2.2. |
| `examples.m` | MATLAB M-file | 9 KB | Executes all examples in the paper. |
| `hypatia` | Text file | 1.4 MB | term-by-document matrix of size 11,390 × 1,265 for Example 4 of [2, Section 5]. |
| `irlba.m` | MATLAB M-file | 23 KB | Implementation of the single-vector-method described in [1]. |
| `irlbablk.m` | MATLAB M-file | 23 KB | Implementation of the block-method described in [2]. |
| `irlbablkdemo.m` | MATLAB M-file | 23 KB | MATLAB demo that illustrates several choices of parameter values; see Table 2.1. |

*Department of Mathematics, University of Rhode Island, Kingston, RI 02881. E-mail: `jbaglama@math.uri.edu`. Home page: http://hypatia.math.uri.edu/∼jbaglama. Research supported in part by NSF grant DMS-0311786.

†Department of Mathematical Sciences, Kent State University, Kent, OH 44242. E-mail: `reichel@math.kent.edu`. Home page: http://www.math.kent.edu/∼reichel. Research supported in part by NSF grant DMS-0107858 and an OBR Research Challenge Grant.

**2. Parameters for `irlba` and `irlbablk`.** The MATLAB functions `irlba` and `irlbablk` require certain parameters to be set. Table 2.1 describes these parameters, their possible values, as well as their default values. The functions use the default values unless a user specifies otherwise.

TABLE 2.1
*Parameters for* `irlba` *and* `irlbablk`.

| | |
|---|---|
| *adjust* | Initial number of vectors added to the $k$ restart vectors. When approximate singular vectors start to converge, more vectors are added to speed up convergence. Default value: $adjust = 3$. |
| *aug* | The value 'RITZ' results in that Ritz vectors are augmented by (block) Krylov subspaces and the value 'HARM' leads to that harmonic Ritz vectors are augmented by the Krylov spaces; see [2]. Default values: $aug$ ='HARM' if $sigma$ ='SS' and $aug$ ='RITZ' if $sigma$ ='LS'. |
| *blsz* | Block-size of Lanczos bidiagonal matrix. The parameter specifies the value of $r$ in [2]. The function `irlba` does not use this parameter. Default value: $blsz = 3$. |
| *disps* | When $disps > 0$, the $k$ desired singular values along with the corresponding residual errors are displayed at each iteration; $disps = 0$ inhibits display of these quantities. Default value: $disps = 0$. |
| *k* | Number of desired singular values or triplets. Default value: $k = 6$. |
| *maxit* | Maximum number of restarts of the (block) Lanczos i bidiagonalization method. Default value: $maxit = 1000$. |
| *m* | Number of steps of the (block) Lanczos bidiagonalization method between restarts at the beginning of the computations. Both `irlba` and `irlbablk` may increase this value during execution to speed up convergence; see [1] for a discussion. A warning message is displayed when the value of $m$ is increased. Default values: $m = 10$ for `irlbablk` and $m = 20$ for `irlba`. |
| *reorth* | Three-letter string specifying whether to use one-sided or two-sided reorthogonalization. Admissible values for the string are:<br>'ONE' - one-sided reorthogonalization.<br>'TWO' - two-sided reorthogonalization.<br>When the matrix $A$ is not square, one-sided reorthogonalization is performed on the "short" vectors. Two-sided reorthogonalization is enforced when a computed estimate of the condition number of $A$ is larger than $1/\sqrt{eps}$, where $eps$ is machine epsilon defined in MATLAB. Default value: $reorth$ ='ONE'. |
| *sigma* | A two-letter string specifying the desired singular values. Admissible values for the string are:<br>'SS' - compute the $k$ smallest singular values,<br>'LS' - compute the $k$ largest singular values.<br>Default value: $sigma$ ='LS'. |
| *tol* | Tolerance used for convergence check. The value of $tol$ specifies the value of $\epsilon$ in [1, 2]. Default value: $tol = 10^{-6}$. |
| *v* | An $n \times r$ matrix of orthonormal starting vectors. The second dimensions must match the block-size, i.e, $r = blsz$ for `irlbablk` and $r = 1$ for `irlba`. By default, the columns of $v$ are determined by orthonormalizing columns generated by the MATLAB function `randn`, which determines a matrix whose entries are random numbers from the standard normal distribution. |

The parameter values for `irlba` and `irlbablk` can be supplied via a structure array with the field values as parameter names. This allows a user to specify and change parameter values easily. The input orders for the parameters are

$$(A, OPTS) \qquad \text{or} \qquad (\text{'Afunc'}, p, n, OPTS),$$

where the first input argument either is a $p \times n$ 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 matrices $A$ and $A^T$. Note that if the first argument is the name of an M-file, then the second and third arguments have to be $p$ and $n$, the number of rows and columns, respectively, of the matrix.

The input argument $OPTS$ is optional. If supplied, then it is a structure array that contains some or all of the input parameters in arbitrary order. For example, the MATLAB command $OPTS.sigma$ ='SS' sets the value of the parameter *sigma* to the letter string 'SS' and, thus, makes `irlba` or `irlbablk` compute the smallest singular values; see Table 2.1 for a description of *sigma* and other input parameters.

TABLE 2.2
*MATLAB M-file* `Afunc.m`

```
function Y = Afunc(X,p,n,blsz,transpose)
   if strcmp(transpose,'F')
      % Compute Y = A*X
      Y=[X(1,:); X(2,:); zeros(p-2,blsz)];
      for i = 3:n
         for r=1:blsz
           Y(i,r) = (i-1)*X(i,r);
         end
      end
   else
      % Compute Y = A'*X
      Y=[X(1,:); X(2,:)];
      for j = 3:n
         for r=1:blsz
           Y(j,r) = (j-1)*X(j,r);
         end
      end
   end
```

The M-file `Afunc.m` for evaluating matrix-vector products with the matrices $A$ and $A^T$ can be written in MATLAB, C, or FORTRAN. In the latter cases, the C or FORTRAN codes are interfaced with MATLAB with the aid of MEX files; see [3]. The M-file `Afunc.m` has the following parameters

$$\begin{aligned} Y &= \text{Afunc}(X, p, n, blsz, TRANS) \quad \text{for} \quad \texttt{irlbablk}, \\ Y &= \text{Afunc}(X, p, n, TRANS) \quad \text{for} \quad \texttt{irlba}. \end{aligned}$$

If $TRANS$ ='F', then $X$ is an $n \times blsz$ matrix and $Y = AX$, where *blsz* specifies the block-size; see Table 2.1. If instead $TRANS$ ='T', then $X$ is an $p \times blsz$ matrix and $Y = A^T X$. For `irlba`, $blsz = 1$ and therefore *blsz* does not have to be provided as a parameter.

A sample MATLAB function `Afunc.m` for `irlbablk` is displayed in Table 2.2. It computes, with the parameters $p = 100$ and $n = 50$, matrix-vector products with the matrices $A$ and $A^T$, where

$$A = \begin{bmatrix} 1 & & & & & 0 \\ & 1 & & & & \\ & & 2 & & & \\ & & & 3 & & \\ & & & & \ddots & \\ & & & & & 49 \\ & & 0 & & & \end{bmatrix} \in \mathbb{R}^{100 \times 50}. \tag{2.1}$$

This `Afunc.m` file is available in the file *na26.tgz*; see Table 1.2.

**3. Output options for `irlba` and `irlbablk`.** There are four available output options. Since the output options are the same for `irlba` and `irlbablk`, we only illustrate them for the latter function.

**a.** `irlbablk`$(A, OPTS)$: Displays the desired singular values.
**b.** $s =$`irlbablk`$(A, OPTS)$: Returns the desired singular values in the vector $s$.
**c.** $[U, S, V] =$`irlbablk`$(A, OPTS)$: Returns the matrices $U$, $S$, and $V$, where $S$ is a diagonal matrix whose diagonal entries are the desired singular values. The singular values are in decreasing order when *sigma*='LS' and in increasing order when *sigma*='SS'. The matrices $U$ and $V$ contain the associated left and right singular vectors, respectively. Thus, $AV = US$, $A^TU = VS$, $V^TV = I$, and $U^TU = I$.
**d.** $[U, S, V, FLAG] =$`irlbablk`$(A, OPTS)$: Returns the matrices $U$, $S$, and $V$ described above, as well as the array $FLAG$ with two entries. $FLAG(1) = 0$ implies normal exit from `irlbablk`: all desired singular values have converged. If $FLAG(1) = 1$, then the maximum number of iterations was reached before all desired singular values were deemed to have converged. In this situation, the matrices $S$, $U$, and $V$ contain the singular triplets that have converged, as well as an approximation of the non-converged singular triplet. The latter is stored in the last columns of $S$, $U$ and $V$. $FLAG(2)$ contains the number of matrix-vector products with $A$ and $A^T$. Here the evaluations of $AX$ and $A^TX$ are counted as $2\,blsz$ matrix-vector products.

**4. Using the MATLAB M-files.** This section present a few calling sequences for the MATLAB functions `examples`, `irlba`, `irlbablk`, and `irlbablkdemo`. All computations presented here were carried out in MATLAB version 6.5. Machine epsilon provided by MATLAB is $eps \approx 2.2 \cdot 10^{-16}$.

**4.1. The function `examples`.** The examples in [2] can be executed by using the M-file `examples.m`. For instance, the command

```
>> examples('example1')
```

yields Example 1 in [2] and plots the graphs of Figure 5.1 in [2]. Acceptable inputs for `examples.m` are `'example1'`, `'example2'`, `'example3'`, or `'example4'`. We remark that the performance depends on the choice of initial (block-) vectors, which are generated randomly. Therefore different runs may yields different timings and output.

**4.2. The graphic user inteface demo `irlbablkdemo`.** The demo can be run with the following command:

```
>> irlbablkdemo
```

The self-explanatory GUI interface displayed in Figure 4.1 appears in a window.

The demo allows a user to choose matrices from the examples of the paper [2], as well as to define other matrices by inserting a MATLAB command, such as "$A =$matrix_definition" at the bottom left-hand side of the GUI.

**4.3. The functions `irlbablk` and `irlba`.** The following MATLAB commands illustrate function calls for `irlba` and `irlbablk`. In the experiments reported below, the matrix $A$ is given by (2.1). This matrix easily can be created by the command

```
>> A=spdiags([1, 1:49]',0,100,50);
```

The command

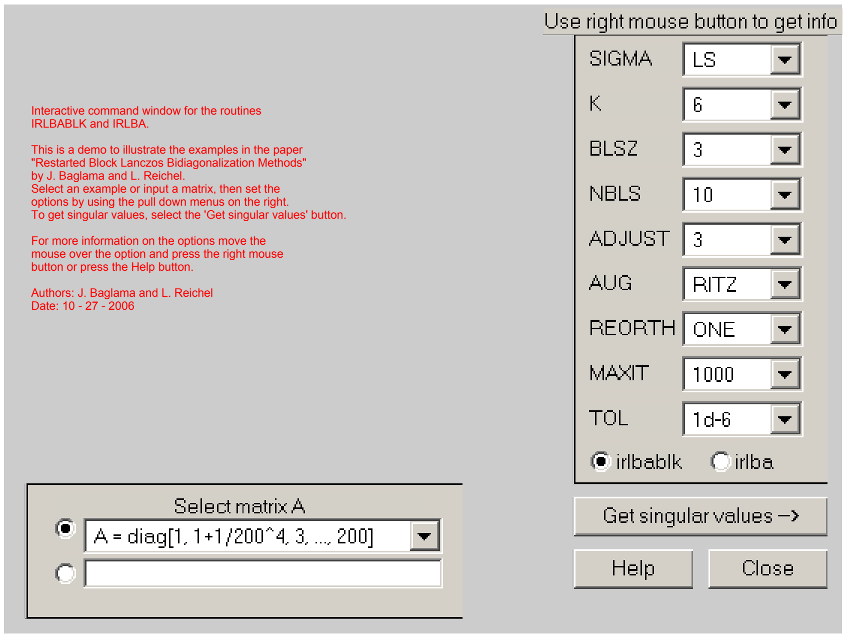
```
>> irlbablk(A)
```

makes `irlbablk` compute the 6 largest singular values of $A$ using the default values for all parameters. The following output is generated

```
SingularValues =

   49.0000
   48.0000
   47.0000
   46.0000
   45.0000
   44.0000
```

FIG. 4.1. *MATLAB demo interface.*



Interactive command window for the routines
IRLBABLK and IRLBA.

This is a demo to illustrate the examples in the paper
"Restarted Block Lanczos Bidiagonalization Methods"
by J. Baglama and L. Reichel.
Select an example or input a matrix, then set the
options by using the pull down menus on the right.
To get singular values, select the 'Get singular values' button.

For more information on the options move the
mouse over the option and press the right mouse
button or press the Help button.

Authors: J. Baglama and L. Reichel
Date: 10 - 27 - 2006

Use right mouse button to get info

| | |
|---|---|
| SIGMA | LS |
| K | 6 |
| BLSZ | 3 |
| NBLS | 10 |
| ADJUST | 3 |
| AUG | RITZ |
| REORTH | ONE |
| MAXIT | 1000 |
| TOL | 1d-6 |

[✓ irlbablk] irlba

Select matrix A

[✓ A = diag[1, 1+1/200^4, 3, ..., 200]]

The command

```
>> irlba(A)
```

results in analogous computations with block-size 1 and yields the same output.

The parameter values can be changed by using the structure array OPTS, whose field values are the parameter names. For example, if we would like to compute the 6 smallest singular values of $A$, then the value of the parameter *sigma* should be set to 'SS'.
This can be achieved by the command

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

Then

```
>> irlbablk(A,OPTS)
```

determines the singular values

```
SingularValues =

    1.0000
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

Using the same structure array OPTS, the command

```
>> irlba(A,OPTS)
```

makes `irbla` seek to determine 6 smallest singular values of $A$. However, `irbla` fails to find the double singular value and produces the output

```
SingularValues =

    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
    6.0000
```

A reduction of the default value for the parameter *tol* is required in order to detect the missed singular value 1. For instance,

```
>> OPTS.tol = eps;
```

sets the tolerance to machine epsilon and the command

```
>> irlba(A,OPTS)
```

now yields the desired output

```
SingularValues =

    1.0000
    1.0000
    2.0000
    3.0000
    4.0000
    5.0000
```

Other parameter values may be changed similarly. For instance, suppose that we would like to determine the 3 smallest singular values by the block Lanczos bidiagonalization method, using block-size 2 with 10 consecutive block-bidiagonalization steps between restarts. Moreover, suppose that we would like to carry out augmentation of harmonic Ritz vectors. This can be achieved by issuing the commands

```
>> OPTS.sigma = 'SS';
>> OPTS.k = 3;
>> OPTS.blsz = 2;
>> OPTS.m = 10;
>> OPTS.aug = 'HARM';
```

The command

```
>> irlbablk(A,OPTS)
```

then produces the output

```
SingularValues =

    1.0000
    1.0000
    2.0000
```

If both singular values and vectors are desired, then `irlbablk` and `irlba` should be called according to

```
>> [U,S,V] = irlbablk(A,OPTS);
```

or

```
>> [U,S,V] = irlba(A,OPTS);
```

respectively.

When the matrix $A$ is implicitly defined by an M-file for the evaluation of matrix-vector products, such as by the file `Afunc.m` of Table 2.2, the call of `irlbablk` should be of the form

```
>> [U,S,V] = irlbablk('Afunc',100,50,OPTS);
```

We remark that the M-file `Afunc.m` of Table 2.2 has to be modified before use with `irlba`, because `irlba` does not use the parameter *blsz*; its value is implicitly assumed to be one. The following simple modification of `Afunc.m` of Table 2.2 gives an M-file `Afunc1.m` that can be used by `irlba`. Rename `Afunc.m` as `Afunc1.m`, remove the parameter *blsz* from the calling sequence, and include as the first assignment $blsz = 1$;. Thus, the first line of `Afunc1.m` reads

```
function Y = Afunc1(X,p,n,transpose)
```

The function `irlba` can then be called as

```
>> [U,S,V] = irlba('Afunc1',100,50,OPTS);
```

## REFERENCES

[1] J. Baglama and L. Reichel, *Augmented implicitly restarted Lanczos bidiagonalization methods*, SIAM J. Sci. Comput., 27 (2005), pp. 19–42.

[2] J. Baglama and L. Reichel, *Restarted block Lanczos bidiagonalization methods*, Numer. Algorithms, in press.

[3] MathWorks, *MATLAB Application Program Interface Guide, Version 5*, Natick, MA, 1998.