ID: A software package for low-rank approximation of matrices via interpolative decompositions, Version 0.4

Per-Gunnar Martinsson, Vladimir Rokhlin,
Yoel Shkolnisky, and Mark Tygert

November 16, 2020

Contents

IMPORTANT

At the minimum, please read Subsection 2.1 and Section 3 below, and beware that the *N.B.*'s in the source code comments highlight key information about the routines; *N.B.* stands for *nota bene* (Latin for "note well").

1 Introduction

This software distribution provides Fortran routines for computing low-rank approximations to matrices, in the forms of interpolative decompositions (IDs) and singular value decompositions (SVDs). The routines use algorithms based on the ID. The ID is also commonly known as the approximation obtained via skeletonization, the approximation obtained via subsampling, and the approximation obtained via subset selection. The ID provides many advantages in many applications, and we suspect that it will become increasingly popular once tools for its computation become more widely available. This software distribution includes some such tools, as well as tools for computing low-rank approximations in the form of SVDs. Section 8 below defines IDs and SVDs, and provides references to detailed discussions of the algorithms used in this software package.

Please beware that normalized power iterations are better suited than the software in this distribution for computing principal component analyses in the typical case when the square of the signal-to-noise ratio is not orders of magnitude greater than both dimensions of the data matrix; see [?].

The algorithms used in this distribution have been optimized for accuracy, efficiency, and reliability; as a somewhat counterintuitive consequence, many must be randomized. All randomized codes in this software package succeed with overwhelmingly high probability (see, for example, [?]). The truly paranoid are welcome to use the routines `idd_diffsnorm` and `idz_diffsnorm` to evaluate rapidly the quality of the approximations produced by the randomized algorithms (as done, for example, in the files `idd_a_test.f`, `idd_r_test.f`, `idz_a_test.f`, and `idz_r_test.f` in the `test` subdirectory of the main directory `id_dist`). In most circumstances, evaluating the quality of an approximation via routines `idd_diffsnorm` or `idz_diffsnorm` is much faster than forming the approximation to be evaluated. Still, we are unaware of any instance in which a properly-compiled routine failed to produce an accurate approximation. To facilitate successful compilation, we encourage the user to read the instructions in the next section, and to read Section 3, too.

2 Compilation instructions

Followed in numerical order, the subsections of this section provide step-by-step instructions for compiling the software under a Unix-compatible operating system.

2.1 Beware that default command-line flags may not be sufficient for compiling the source codes!

The Fortran source codes in this distribution pass `real*8` variables as integer variables, integers as `real*8`'s, `real*8`'s as `complex*16`'s, and so on. This is common practice in numerical codes, and is not an error; be sure to provide the relevant command-line flags to the compiler (for example, run `fort77` and `f2c` with the flag `-!P`). When following the compilation instructions in Subsection 2.4 below, be sure to set `FFLAGS` appropriately.

2.2 Install LAPACK

The SVD routines in this distribution depend on LAPACK. Before compiling the present distribution, create the LAPACK and BLAS archive (library) `.a` files; information about installing LAPACK is available at `http://www.netlib.org/lapack/` (and several other web sites).

2.3 Decompress and untar the file `id_dist.tar.gz`

At the command line, decompress and untar the file `id_dist.tar.gz` by issuing a command such as `tar -xvvzf id_dist.tar.gz`. This will create a directory named `id_dist`.

2.4 Edit the Makefile

The directory `id_dist` contains a file named `Makefile`. In `Makefile`, set the following:

- `FC` is the Fortran compiler.
- `FFLAGS` is the set of command-line flags (specifying optimization settings, for example) for the Fortran compiler specified by `FC`; please heed the warning in Subsection 2.1 above!
- `BLAS_LIB` is the file-system path to the BLAS archive (library) `.a` file.
- `LAPACK_LIB` is the file-system path to the LAPACK archive (library) `.a` file.
- `ARCH` is the archiver utility (usually `ar`).
- `ARCHFLAGS` is the set of command-line flags for the archiver specified by `ARCH` needed to create an archive (usually `cr`).
- `RANLIB` is to be set to `ranlib` when `ranlib` is available, and is to be set to `echo` when `ranlib` is not available.

2.5 Make and test the libraries

At the command line in a shell that adheres to the Bourne shell conventions for redirection, issue the command "`make clean; make`" to both create the archive (library) `id_lib.a` and test it. (In most modern Unix distributions, `sh` is the Bourne shell, or else is fully compatible with the Bourne shell; the Korn shell `ksh` and the Bourne-again shell `bash` also use the Bourne shell conventions for redirection.) `make` places the file `id_lib.a` in the directory `id_dist`; the archive (library) file `id_lib.a` contains machine code for all user-callable routines in this distribution.

3 Naming conventions

The names of routines and files in this distribution start with prefixes, followed by an underscore ("_"). The prefixes are two to four characters in length, and have the following meanings:

- The first two letters are always "`id`", the name of this distribution.
- The third letter (when present) is either "`d`" or "`z`"; "`d`" stands for double precision (`real*8`), and "`z`" stands for double complex (`complex*16`).
- The fourth letter (when present) is either "`r`" or "`p`"; "`r`" stands for specified rank, and "`p`" stands for specified precision. The specified rank routines require the user to provide the rank of the approximation to be constructed, while the specified precision routines adjust the rank adaptively to attain the desired precision.

For example, `iddr_aid` is a `real*8` routine which computes an approximation of specified rank. `idz_snorm` is a `complex*16` routine. `id_randperm` is yet another routine in this distribution.

4 Example programs

For examples of how to use the user-callable routines in this distribution, see the source codes in subdirectory `test` of the main directory `id_dist`.

5 Directory structure

The main `id_dist` directory contains a Makefile, the auxiliary text files `README.txt` and `size.txt`, and the following subdirectories, described in the subsections below:

1. `bin`
2. `development`
3. `doc`

4. `src`
5. `test`
6. `tmp`

If a "`make all`" command has completed successfully, then the main `id_dist` directory will also contain an archive (library) file `id_lib.a` containing machine code for all of the user-callable routines.

5.1 Subdirectory `bin`

Once all of the libraries have been made via the Makefile in the main `id_dist` directory, the subdirectory `bin` will contain object files (machine code), each compiled from the corresponding file of source code in the subdirectory `src` of `id_dist`.

5.2 Subdirectory `development`

Each Fortran file in the subdirectory `development` (except for `dfft.f` and `prini.f`) specifies its dependencies at the top, then provides a main program for testing and debugging, and finally provides source code for a library of user-callable subroutines. The Fortran file `dfft.f` is a copy of P. N. Swarztrauber's FFTPACK library for computing fast Fourier transforms. The Fortran file `prini.f` is a copy of V. Rokhlin's library of formatted printing routines. Both `dfft.f` (version 4) and `prini.f` are in the public domain. The shell script `RUNME.sh` runs shell scripts `make_src.sh` and `make_test.sh`, which fill the subdirectories `src` and `test` of the main directory `id_dist` with source codes for user-callable routines and with the main program testing codes.

5.3 Subdirectory `doc`

Subdirectory `doc` contains this documentation, supplementing comments in the source codes.

5.4 Subdirectory `src`

The files in the subdirectory `src` provide source code for software libraries. Each file in the subdirectory `src` (except for `dfft.f` and `prini.f`) is the bottom part of the corresponding file in the subdirectory `development` of `id_dist`. The file `dfft.f` is just a copy of P. N. Swarztrauber's FFTPACK library for computing fast Fourier transforms. The file `prini.f` is a copy of V. Rokhlin's library of formatted printing routines. Both `dfft.f` (version 4) and `prini.f` are in the public domain.

5.5 Subdirectory `test`

The files in subdirectory `test` provide source code for testing and debugging. Each file in subdirectory `test` is the top part of the corresponding file in subdirectory `development` of `id_dist`, and provides a main program and a list of its dependencies. These codes provide examples of how to call the user-callable routines.

6 Catalog of the routines

The main routines for decomposing `real*8` matrices are:

1. IDs of arbitrary (generally dense) matrices: `iddp_id`, `iddr_id`, `iddp_aid`, `iddr_aid`
2. IDs of matrices that may be rapidly applied to arbitrary vectors (as may the matrices' transposes): `iddp_rid`, `iddr_rid`
3. SVDs of arbitrary (generally dense) matrices: `iddp_svd`, `iddr_svd`, `iddp_asvd`, `iddr_asvd`
4. SVDs of matrices that may be rapidly applied to arbitrary vectors (as may the matrices' transposes): `iddp_rsvd`, `iddr_rsvd`

Similarly, the main routines for decomposing `complex*16` matrices are:

1. IDs of arbitrary (generally dense) matrices: `idzp_id`, `idzr_id`, `idzp_aid`, `idzr_aid`
2. IDs of matrices that may be rapidly applied to arbitrary vectors (as may the matrices' adjoints): `idzp_rid`, `idzr_rid`
3. SVDs of arbitrary (generally dense) matrices: `idzp_svd`, `idzr_svd`, `idzp_asvd`, `idzr_asvd`
4. SVDs of matrices that may be rapidly applied to arbitrary vectors (as may the matrices' adjoints): `idzp_rsvd`, `idzr_rsvd`

This distribution also includes routines for constructing pivoted QR decompositions (in `idd_qrpiv.f` and `idz_qrpiv.f`), for estimating the spectral norms of matrices that may be applied rapidly to arbitrary vectors as may their adjoints (in `idd_snorm.f` and `idz_snorm.f`), for converting IDs to SVDs (in `idd_id2svd.f` and `idz_id2svd.f`), and for computing rapidly arbitrary subsets of the entries of the discrete Fourier transforms of vectors (in `idd_sfft.f` and `idz_sfft.f`).

6.1 List of the routines

The following is an alphabetical list of the routines in this distribution, together with brief descriptions of their functionality and the names of the files containing the routines' source code:

Routine	**Description**	**Source file**
`id_frand`	generates pseudorandom numbers drawn uniformly from the interval $[0, 1]$; this routine is more efficient than routine `id_srand`, but cannot generate fewer than 55 pseudorandom numbers per call	`id_rand.f`

Routine	**Description**	**Source file**
`id_frandi`	initializes the seed values for routine `id_frand` to specified values	`id_rand.f`
`id_frando`	initializes the seed values for routine `id_frand` to their original, default values	`id_rand.f`
`id_randperm`	generates a uniformly random permutation	`id_rand.f`
`id_srand`	generates pseudorandom numbers drawn uniformly from the interval $[0, 1]$; this routine is less efficient than routine `id_frand`, but can generate fewer than 55 pseudorandom numbers per call	`id_rand.f`
`id_srandi`	initializes the seed values for routine `id_srand` to specified values	`id_rand.f`
`id_srando`	initializes the seed values for routine `id_srand` to their original, default values	`id_rand.f`
`idd_copycols`	collects together selected columns of a matrix	`idd_id.f`
`idd_diffsnorm`	estimates the spectral norm of the difference between two matrices specified by routines for applying the matrices and their transposes to arbitrary vectors; this routine uses the power method with a random starting vector	`idd_snorm.f`
`idd_enorm`	calculates the Euclidean norm of a vector	`idd_snorm.f`
`idd_estrank`	estimates the numerical rank of an arbitrary (generally dense) matrix to a specified precision; this routine is randomized, and must be initialized with routine `idd_frmi`	`iddp_aid.f`
`idd_frm`	transforms a vector into a vector which is sufficiently scrambled to be subsampled, via a composition of Rokhlin's random transform, random subselection, and a fast Fourier transform	`idd_frm.f`
`idd_frmi`	initializes routine `idd_frm`	`idd_frm.f`
`idd_getcols`	collects together selected columns of a matrix specified by a routine for applying the matrix to arbitrary vectors	`idd_id.f`
`idd_house`	calculates the vector and scalar needed to apply the Householder transformation reflecting a given vector into its first entry	`idd_house.f`
`idd_houseapp`	applies a Householder matrix to a vector	`idd_house.f`
`idd_id2svd`	converts an approximation to a matrix in the form of an ID into an approximation in the form of an SVD	`idd_id2svd.f`

Routine	**Description**	**Source file**
`idd_ldiv`	finds the greatest integer less than or equal to a specified integer, that is divisible by another (larger) specified integer	`idd_sfft.f`
`idd_pairsamps`	calculates the indices of the pairs of integers that the individual integers in a specified set belong to	`idd_frm.f`
`idd_permmult`	multiplies together a bunch of permutations	`idd_qrpiv.f`
`idd_qinqr`	reconstructs the Q matrix in a QR decomposition from the output of routines `iddp_qrpiv` or `iddr_qrpiv`	`idd_qrpiv.f`
`idd_qrmatmat`	applies to multiple vectors collected together as a matrix the Q matrix (or its transpose) in the QR decomposition of a matrix, as described by the output of routines `iddp_qrpiv` or `iddr_qrpiv`; to apply Q (or its transpose) to a single vector without having to provide a work array, use routine `idd_qrmatvec` instead	`idd_qrpiv.f`
`idd_qrmatvec`	applies to a single vector the Q matrix (or its transpose) in the QR decomposition of a matrix, as described by the output of routines `iddp_qrpiv` or `iddr_qrpiv`; to apply Q (or its transpose) to several vectors efficiently, use routine `idd_qrmatmat` instead	`idd_qrpiv.f`
`idd_random_transf`	applies rapidly a random orthogonal matrix to a user-supplied vector	`id_rtrans.f`
`idd_random_transf_init`	initializes routines `idd_random_transf` and `idd_random_transf_inverse`	`id_rtrans.f`
`idd_random_transf_inverse`	applies rapidly the inverse of the operator applied by routine `idd_random_transf`	`id_rtrans.f`
`idd_reconid`	reconstructs a matrix from its ID	`idd_id.f`
`idd_reconint`	constructs P in the ID $A = B\,P$, where the columns of B are a subset of the columns of A, and P is the projection coefficient matrix, given `list`, `krank`, and `proj` output by routines `iddr_id`, `iddp_id`, `iddr_aid`, `iddp_aid`, `iddr_rid`, or `iddp_rid`	`idd_id.f`

Routine	**Description**	**Source file**
`idd_sfft`	rapidly computes a subset of the entries of the discrete Fourier transform of a vector, composed with permutation matrices both on input and on output	`idd_sfft.f`
`idd_sffti`	initializes routine `idd_sfft`	`idd_sfft.f`
`idd_sfrm`	transforms a vector into a scrambled vector of specified length, via a composition of Rokhlin's random transform, random subselection, and a fast Fourier transform	`idd_frm.f`
`idd_sfrmi`	initializes routine `idd_sfrm`	`idd_frm.f`
`idd_snorm`	estimates the spectral norm of a matrix specified by routines for applying the matrix and its transpose to arbitrary vectors; this routine uses the power method with a random starting vector	`idd_snorm.f`
`iddp_aid`	computes the ID of an arbitrary (generally dense) matrix, to a specified precision; this routine is randomized, and must be initialized with routine `idd_frmi`	`iddp_aid.f`
`iddp_asvd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified precision; this routine is randomized, and must be initialized with routine `idd_frmi`	`iddp_asvd.f`
`iddp_id`	computes the ID of an arbitrary (generally dense) matrix, to a specified precision; this routine is often less efficient than routine `iddp_aid`	`idd_id.f`
`iddp_qrpiv`	computes the pivoted QR decomposition of an arbitrary (generally dense) matrix via Householder transformations, stopping at a specified precision of the decomposition	`idd_qrpiv.f`
`iddp_rid`	computes the ID, to a specified precision, of a matrix specified by a routine for applying its transpose to arbitrary vectors; this routine is randomized	`iddp_rid.f`
`iddp_rsvd`	computes the SVD, to a specified precision, of a matrix specified by routines for applying the matrix and its transpose to arbitrary vectors; this routine is randomized	`iddp_rsvd.f`

Routine	Description	Source file
`iddp_svd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified precision; this routine is often less efficient than routine `iddp_asvd`	`idd_svd.f`
`iddr_aid`	computes the ID of an arbitrary (generally dense) matrix, to a specified rank; this routine is randomized, and must be initialized by routine `iddr_aidi`	`iddr_aid.f`
`iddr_aidi`	initializes routine `iddr_aid`	`iddr_aid.f`
`iddr_asvd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified rank; this routine is randomized, and must be initialized with routine `idd_aidi`	`iddr_asvd.f`
`iddr_id`	computes the ID of an arbitrary (generally dense) matrix, to a specified rank; this routine is often less efficient than routine `iddr_aid`	`idd_id.f`
`iddr_qrpiv`	computes the pivoted QR decomposition of an arbitrary (generally dense) matrix via Householder transformations, stopping at a specified rank of the decomposition	`idd_qrpiv.f`
`iddr_rid`	computes the ID, to a specified rank, of a matrix specified by a routine for applying its transpose to arbitrary vectors; this routine is randomized	`iddr_rid.f`
`iddr_rsvd`	computes the SVD, to a specified rank, of a matrix specified by routines for applying the matrix and its transpose to arbitrary vectors; this routine is randomized	`iddr_rsvd.f`
`iddr_svd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified rank; this routine is often less efficient than routine `iddr_asvd`	`idd_svd.f`
`idz_copycols`	collects together selected columns of a matrix	`idz_id.f`
`idz_diffsnorm`	estimates the spectral norm of the difference between two matrices specified by routines for applying the matrices and their adjoints to arbitrary vectors; this routine uses the power method with a random starting vector	`idz_snorm.f`
`idz_enorm`	calculates the Euclidean norm of a vector	`idz_snorm.f`
`idz_estrank`	estimates the numerical rank of an arbitrary (generally dense) matrix to a specified precision; this routine is randomized, and must be initialized with routine `idz_frmi`	`idzp_aid.f`

Routine	**Description**	**Source file**
`idz_frm`	transforms a vector into a vector which is sufficiently scrambled to be subsampled, via a composition of Rokhlin's random transform, random subselection, and a fast Fourier transform	`idz_frm.f`
`idz_frmi`	initializes routine `idz_frm`	`idz_frm.f`
`idz_getcols`	collects together selected columns of a matrix specified by a routine for applying the matrix to arbitrary vectors	`idz_id.f`
`idz_house`	calculates the vector and scalar needed to apply the Householder transformation reflecting a given vector into its first entry	`idz_house.f`
`idz_houseapp`	applies a Householder matrix to a vector	`idz_house.f`
`idz_id2svd`	converts an approximation to a matrix in the form of an ID into an approximation in the form of an SVD	`idz_id2svd.f`
`idz_ldiv`	finds the greatest integer less than or equal to a specified integer, that is divisible by another (larger) specified integer	`idz_sfft.f`
`idz_permmult`	multiplies together a bunch of permutations	`idz_qrpiv.f`
`idz_qinqr`	reconstructs the Q matrix in a QR decomposition from the output of routines `idzp_qrpiv` or `idzr_qrpiv`	`idz_qrpiv.f`
`idz_qrmatmat`	applies to multiple vectors collected together as a matrix the Q matrix (or its adjoint) in the QR decomposition of a matrix, as described by the output of routines `idzp_qrpiv` or `idzr_qrpiv`; to apply Q (or its adjoint) to a single vector without having to provide a work array, use routine `idz_qrmatvec` instead	`idz_qrpiv.f`
`idz_qrmatvec`	applies to a single vector the Q matrix (or its adjoint) in the QR decomposition of a matrix, as described by the output of routines `idzp_qrpiv` or `idzr_qrpiv`; to apply Q (or its adjoint) to several vectors efficiently, use routine `idz_qrmatmat` instead	`idz_qrpiv.f`

Routine	**Description**	**Source file**
`idz_random_transf`	applies rapidly a random unitary matrix to a user-supplied vector	`id_rtrans.f`
`idz_random_transf_init`	initializes routines `idz_random_transf` and `idz_random_transf_inverse`	`id_rtrans.f`
`idz_random_transf_inverse`	applies rapidly the inverse of the operator applied by routine `idz_random_transf`	`id_rtrans.f`
`idz_reconid`	reconstructs a matrix from its ID	`idz_id.f`
`idz_reconint`	constructs P in the ID $A = B\,P$, where the columns of B are a subset of the columns of A, and P is the projection coefficient matrix, given `list`, `krank`, and `proj` output by routines `idzr_id`, `idzp_id`, `idzr_aid`, `idzp_aid`, `idzr_rid`, or `idzp_rid`	`idz_id.f`
`idz_sfft`	rapidly computes a subset of the entries of the discrete Fourier transform of a vector, composed with permutation matrices both on input and on output	`idz_sfft.f`
`idz_sffti`	initializes routine `idz_sfft`	`idz_sfft.f`
`idz_sfrm`	transforms a vector into a scrambled vector of specified length, via a composition of Rokhlin's random transform, random subselection, and a fast Fourier transform	`idz_frm.f`
`idz_sfrmi`	initializes routine `idz_sfrm`	`idz_frm.f`
`idz_snorm`	estimates the spectral norm of a matrix specified by routines for applying the matrix and its adjoint to arbitrary vectors; this routine uses the power method with a random starting vector	`idz_snorm.f`
`idzp_aid`	computes the ID of an arbitrary (generally dense) matrix, to a specified precision; this routine is randomized, and must be initialized with routine `idz_frmi`	`idzp_aid.f`
`idzp_asvd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified precision; this routine is randomized, and must be initialized with routine `idz_frmi`	`idzp_asvd.f`
`idzp_id`	computes the ID of an arbitrary (generally dense) matrix, to a specified precision; this routine is often less efficient than routine `idzp_aid`	`idz_id.f`

Routine	Description	Source file
`idzp_qrpiv`	computes the pivoted QR decomposition of an arbitrary (generally dense) matrix via Householder transformations, stopping at a specified precision of the decomposition	`idz_qrpiv.f`
`idzp_rid`	computes the ID, to a specified precision, of a matrix specified by a routine for applying its adjoint to arbitrary vectors; this routine is randomized	`idzp_rid.f`
`idzp_rsvd`	computes the SVD, to a specified precision, of a matrix specified by routines for applying the matrix and its adjoint to arbitrary vectors; this routine is randomized	`idzp_rsvd.f`
`idzp_svd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified precision; this routine is often less efficient than routine `idzp_asvd`	`idz_svd.f`
`idzr_aid`	computes the ID of an arbitrary (generally dense) matrix, to a specified rank; this routine is randomized, and must be initialized by routine `idzr_aidi`	`idzr_aid.f`
`idzr_aidi`	initializes routine `idzr_aid`	`idzr_aid.f`
`idzr_asvd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified rank; this routine is randomized, and must be initialized with routine `idz_aidi`	`idzr_asvd.f`
`idzr_id`	computes the ID of an arbitrary (generally dense) matrix, to a specified rank; this routine is often less efficient than routine `idzr_aid`	`idz_id.f`
`idzr_qrpiv`	computes the pivoted QR decomposition of an arbitrary (generally dense) matrix via Householder transformations, stopping at a specified rank of the decomposition	`idz_qrpiv.f`
`idzr_rid`	computes the ID, to a specified rank, of a matrix specified by a routine for applying its adjoint to arbitrary vectors; this routine is randomized	`idzr_rid.f`
`idzr_rsvd`	computes the SVD, to a specified rank, of a matrix specified by routines for applying the matrix and its adjoint to arbitrary vectors; this routine is randomized	`idzr_rsvd.f`

Routine	Description	Source file
`idzr_svd`	computes the SVD of an arbitrary (generally dense) matrix, to a specified rank; this routine is often less efficient than routine `idzr_asvd`	`idz_svd.f`

7 Documentation in the source codes

Each routine in the source codes includes documentation in the comments immediately following the declaration of the subroutine's calling sequence. This documentation describes the purpose of the routine, the input and output variables, and the required work arrays (if any). This documentation also cites relevant references. Please pay attention to the *N.B.*'s; *N.B.* stands for *nota bene* (Latin for "note well") and highlights important information about the routines.

8 Notation and decompositions

This section sets notational conventions employed in this documentation and the associated software, and defines both the singular value decomposition (SVD) and the interpolative decomposition (ID). For information concerning other mathematical objects used in the code (such as Householder transformations, pivoted QR decompositions, and discrete and fast Fourier transforms — DFTs and FFTs), see, for example, [?]. For detailed descriptions and proofs of the mathematical facts discussed in the present section, see, for example, [?] and the references in [?].

Throughout this document and the accompanying software distribution, $\|\mathbf{x}\|$ always denotes the Euclidean norm of the vector $\mathbf{x}$, and $\|A\|$ always denotes the spectral norm of the matrix A. Subsection 8.1 below defines the Euclidean norm; Subsection 8.2 below defines the spectral norm. We use A^* to denote the adjoint of the matrix A.

8.1 Euclidean norm

For any positive integer n, and vector $\mathbf{x}$ of length n, the Euclidean (l^2) norm $\|\mathbf{x}\|$ is

$$\|\mathbf{x}\| = \sqrt{\sum_{k=1}^{n} |x_k|^2}, \tag{1}$$

where x_1, x_2, …, x_{n-1}, x_n are the entries of $\mathbf{x}$.

8.2 Spectral norm

For any positive integers m and n, and $m \times n$ matrix A, the spectral (l^2 operator) norm $\|A\|$ is

$$\|A_{m\times n}\| = \max \frac{\|A_{m\times n}\,\mathbf{x}_{n\times 1}\|}{\|\mathbf{x}_{n\times 1}\|}, \tag{2}$$

where the max is taken over all $n \times 1$ column vectors $\mathbf{x}$ such that $\|\mathbf{x}\| \neq 0$.

8.3 Singular value decomposition (SVD)

For any positive real number ε, positive integers k, m, and n with $k \le m$ and $k \le n$, and any $m \times n$ matrix A, a rank-k approximation to A in the form of an SVD (to precision ε) consists of an $m \times k$ matrix U whose columns are orthonormal, an $n \times k$ matrix V whose columns are orthonormal, and a diagonal $k \times k$ matrix Σ with diagonal entries $\Sigma_{1,1} \ge \Sigma_{2,2} \ge \cdots \ge \Sigma_{n-1,n-1} \ge \Sigma_{n,n} \ge 0$, such that

$$\|A_{m\times n} - U_{m\times k}\,\Sigma_{k\times k}\,(V^*)_{k\times n}\| \le \varepsilon. \tag{3}$$

The product $U\,\Sigma\,V^*$ is known as an SVD. The columns of U are known as left singular vectors; the columns of V are known as right singular vectors. The diagonal entries of Σ are known as singular values.

When $k = m$ or $k = n$, and $A = U\,\Sigma\,V^*$, then $U\,\Sigma\,V^*$ is known as the SVD of A; the columns of U are the left singular vectors of A, the columns of V are the right singular vectors of A, and the diagonal entries of Σ are the singular values of A. For any positive integer k with $k < m$ and $k < n$, there exists a rank-k approximation to A in the form of an SVD, to precision σ_{k+1}, where σ_{k+1} is the $(k+1)^{\text{st}}$ greatest singular value of A.

8.4 Interpolative decomposition (ID)

For any positive real number ε, positive integers k, m, and n with $k \le m$ and $k \le n$, and any $m \times n$ matrix A, a rank-k approximation to A in the form of an ID (to precision ε) consists of a $k \times n$ matrix P, and an $m \times k$ matrix B whose columns constitute a subset of the columns of A, such that

1. $\|A_{m\times n} - B_{m\times k}\,P_{k\times n}\| \le \varepsilon$,
2. some subset of the columns of P makes up the $k \times k$ identity matrix, and
3. every entry of P has an absolute value less than or equal to a reasonably small positive real number, say 2.

The product $B\,P$ is known as an ID. The matrix P is known as the projection or interpolation matrix of the ID. Property 1 above approximates each column of A via a linear combination of the columns of B (which are themselves columns of A), with the coefficients in the linear combination given by the entries of P.

The interpolative decomposition is "interpolative" due to Property 2 above. The ID is numerically stable due to Property 3 above. It follows from Property 2 that the least (k^{th} greatest) singular value of P is at least 1. Combining Properties 2 and 3 yields that

$$\|P_{k\times n}\| \le \sqrt{4k(n-k)+1}. \tag{4}$$

When $k = m$ or $k = n$, and $A = B\,P$, then $B\,P$ is known as the ID of A. For any positive integer k with $k < m$ and $k < n$, there exists a rank-k approximation to A in the form of an ID, to precision $\sqrt{k(n-k)+1}\;\sigma_{k+1}$, where σ_{k+1} is the $(k+1)^{\text{st}}$ greatest singular value of A (in fact, there exists an ID in which every entry of the projection matrix P has an absolute value less than or equal to 1).

9 Bug reports, feedback, and support

Please let us know about errors in the software or in the documentation via e-mail to `tygert@aya.yale.edu`. We would also appreciate hearing about particular applications of the codes, especially in the form of journal articles e-mailed to `tygert@aya.yale.edu`. Mathematical and technical support may also be available via e-mail. Enjoy!