lapack_wrapper

NAME

lapack_wrapper - a frovedis module provides user-friendly interfaces for commonly used lapack routines in scientific applications like machine learning algorithms.

SYNOPSIS

```
#include <frovedis/matrix/lapack_wrapper.hpp>
```

WRAPPER FUNCTIONS

int getrf (const `sliced_colmajor_matrix_local<T>`& m,
`std::vector<int>`& ipiv)
int getri (const `sliced_colmajor_matrix_local<T>`& m,
const `std::vector<int>`& ipiv)
int getrs (const `sliced_colmajor_matrix_local<T>`& m1,
const `sliced_colmajor_matrix_local<T>`& m2,
const `std::vector<int>`& ipiv,
char trans = 'N')
int gesv (const `sliced_colmajor_matrix_local<T>`& m1,
const `sliced_colmajor_matrix_local<T>`& m2)
int gesv (const `sliced_colmajor_matrix_local<T>`& m1,
const `sliced_colmajor_matrix_local<T>`& m2,
`std::vector<int>`& ipiv)
int gels (const `sliced_colmajor_matrix_local<T>`& m1,
const `sliced_colmajor_matrix_local<T>`& m2,
char trans = 'N')
int gesvd (const `sliced_colmajor_matrix_local<T>`& m,
`std::vector<T>`& sval,
char option = 'N')
int gesvd (const `sliced_colmajor_matrix_local<T>`& m,
`std::vector<T>`& sval,
const `sliced_colmajor_matrix_local<T>`& svec,
char vtype = 'L',
char part = 'A',
char opt_a = 'N')
int gesvd (const `sliced_colmajor_matrix_local<T>`& m,
`std::vector<T>`& sval,
const `sliced_colmajor_matrix_local<T>`& lsvec,
const `sliced_colmajor_matrix_local<T>`& rsvec,

char part_l = 'A',
char part_r = 'A')
int gesdd (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& sval)
int gesdd (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& sval,
const sliced_colmajor_matrix_local<T>& svec)
int gesdd (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& sval,
const sliced_colmajor_matrix_local<T>& lsvec,
const sliced_colmajor_matrix_local<T>& rsvec,
char part_lr = 'A')
int gelsy (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
T rcond = -1)
int gelsy (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
int& rank,
T rcond = -1)
int gelss (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
T rcond = -1)
int gelss (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
std::vector<T>& sval,
int& rank,
T rcond = -1)
int gelsd (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
T rcond = -1)
int gelsd (const sliced_colmajor_matrix_local<T>& m1,
const sliced_colmajor_matrix_local<T>& m2,
std::vector<T>& sval,
int& rank,
T rcond = -1)
int geev (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& eval)
int geev (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& eval,
const sliced_colmajor_matrix_local<T>& evec,
char vtype = 'L')
int geev (const sliced_colmajor_matrix_local<T>& m,
std::vector<T>& eval,
const sliced_colmajor_matrix_local<T>& levec,
const sliced_colmajor_matrix_local<T>& revec)

SPECIAL FUNCTIONS

colmajor_matrix_local<T> inv (const sliced_colmajor_matrix_local<T>& m)

DESCRIPTION

LAPACK is a high-performance external library written in Fortran language. It provides rich set of linear algebra functionalities. Like ScaLAPACK, computation loads of these functionalities **are not parallelized** over the available processes in a system, thus they operate on *non-distributed* data. But like ScaLAPACK, the user interfaces of this library are also very detailed and a bit complex in nature. It requires a strong understanding on each of the input parameters before using these functionalities correctly.

Frovedis provides a wrapper module for some commonly used LAPACK subroutines in scientific applications like machine learning algorithms. These wrapper interfaces are very simple and user needs not to consider all the detailed input parameters. Only specifying the target vectors or matrices with some other parameters (depending upon need) are fine. At the same time, all the use cases of a LAPACK routine can also be performed using Frovedis LAPACK wrapper of that routine.

These wrapper routines are global functions in nature. Thus they can be called easily from within the "frovedis" namespace. As an input matrix, they accept "`colmajor_matrix_local<T>`" or "`sliced_colmajor_matrix_local<T>`". "T" is a template type which can be either "float" or "double". The individual detailed descriptions can be found in the subsequent sections. Please note that the term "inout", used in the below section indicates a function argument as both "input" and "output".

Detailed Description

getrf (m, ipiv)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
ipiv: An empty object of the type `std::vector<int>` (output)

Purpose
It computes an LU factorization of a general M-by-N matrix, "m" using partial pivoting with row interchanges.

On successful factorization, matrix "m" is overwritten with the computed L and U factors. Along with the input matrix, this function expects user to pass an empty object of the type "`std::vector<int>`" as a second argument, named as "ipiv" which would be updated with the pivoting information associated with input matrix "m" by this function while computing factors. This "ipiv" information will be useful in computation of some other functions (like getri, getrs etc.)

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

getri (m, ipiv)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
ipiv: An object of the type `std::vector<int>` (input)

Purpose
It computes the inverse of a square matrix using the LU factorization computed by getrf(). So in order to compute inverse of a matrix, first compute it's LU factor (and ipiv information) using getrf() and then pass the factored matrix, "m" along with the "ipiv" information to this function.

On success, factored matrix "m" is overwritten with the inverse (of the matrix which was passed to getrf()) matrix. "ipiv" will be internally used by this function and will remain unchanged.

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

getrs (m1, m2, ipiv, trans='N')

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (input)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
ipiv: An object of the type `std::vector<int>` (input)
trans: A character containing either 'N' or 'T' [Default: 'N'] (input/optional)

Purpose
It solves a real system of linear equations, AX=B with a general square matrix (A) using the LU factorization computed by getrf(). Thus before calling this function, it is required to obtain the factored matrix "m1" (along with "ipiv" information) by calling getrf().

If trans='N', the linear equation AX=B is solved.
If trans='T' the linear equation transpose(A)X=B (A'X=B) is solved.

The matrix "m2" should have number of rows >= the number of rows in "m1" and at least 1 column in it.

On entry, "m2" contains the right-hand-side (B) of the equation and on successful exit it is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesv (m1, m2)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)

Purpose
It solves a real system of linear equations, AX=B with a general square matrix, "m1" by computing it's LU factors internally. This function internally computes the LU factors and ipiv information using getrf() and then solves the equation using getrs().

The matrix "m2" should have number of rows >= the number of rows in "m1" and at least 1 column in it.

On entry, "m1" contains the left-hand-side square matrix (A), "m2" contains the right-hand-side matrix (B) and on successful exit "m1" is overwritten with it's LU factors, "m2" is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesv (m1, m2, ipiv)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
ipiv: An empty object of the type `std::vector<int>` (output)

Purpose
The function serves the same purpose as explained in above version of gesv (with two parameters). Only difference is that this version accepts an extra parameter "ipiv" of the type `std::vector<int>` which would be allocated and updated with the pivoting information computed during factorization of "m1". Along with the factored matrix, it might also be needed to know the associated pivot values. In that case, this version of gesv (with three parameters) can be used.

On entry, "m1" contains the left-hand-side square matrix (A), "m2" contains the right-hand-side matrix (B), and "ipiv" is an empty object. On successful exit "m1" is overwritten with it's LU factors, "m2" is overwritten with the solution matrix (X), and "ipiv" is updated with the pivot values associated with factored matrix, "m1".

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gels (m1, m2, trans='N')

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (input)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
trans: A character containing either 'N' or 'T' [Default: 'N'] (input/optional)

Purpose
It solves overdetermined or underdetermined real linear systems involving an M-by-N matrix (A) or its transpose, using a QR or LQ factorization of (A). It is assumed that matrix (A) has full rank.

If trans='N' and M >= N: it finds the least squares solution of an overdetermined system.
If trans='N' and M < N: it finds the minimum norm solution of an underdetermined system.
If trans='T' and M >= N: it finds the minimum norm solution of an underdetermined system.
If trans='T' and M < N: it finds the least squares solution of an overdetermined system.

The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, "m1" is overwritten with the QR or LQ factors and "m2" is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesvd (m, sval, option='N')

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)
option: A character containing either 'L', 'R' or 'N' [Default: 'N'] (input/optional)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. Optionally, it can also compute part of left or right singular vectors.

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`. And on exit, if option='L', then "m" is overwritten with the first min(M,N) columns of left singular vectors (stored columnwise).
If option='R', then "m" is overwritten with the first min(M,N) rows of right singular vectors (stored rowwise

in transposed form).
And if option='N', neither right nor left singular vectors are computed and the contents of "m" is destroyed (used as workspace internally by this function).
"sval" is updated with the singular values in sorted oder, so that sval(i) >= sval(i+1).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesvd (m, sval, svec, vtype='L', part='A', opt_a='N')

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)
svec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
vtype: A character value containing either 'L' or 'R' [Default: 'L'] (input/optional)
part: A character value containing either 'A' or 'S' [Default: 'A'] (input/optional)
opt_a: A character value containing either 'O' or 'N' [Default: 'N'] (input/optional)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. Additionally, it also computes *left and/or right singular vectors*.

If vtype='L' and part='A', "svec" will be updated with all the M columns of left singular vectors. In that case, "svec" should have at least M number of rows and M number of columns.
If vtype='L' and part='S', "svec" will be updated with first min(M,N) columns of left singular vectors (stored columnwise). In that case, "svec" should have at least M number of rows and min(M,N) number of columns.
If vtype='R' and part='A', "svec" will be updated with all the N rows of right singular vectors (in transposed form). In that case, "svec" should have at least N number of rows and N number of columns.
If vtype='R' and part='S', "svec" will be updated with first min(M,N) rows of right singular vectors (stored rowwise in transposed form). In that case, "svec" should have at least min(M,N) number of rows and N number of columns.

This function expects that required memory would be allocated for the output matrix "svec" beforehand. If it is not allocated, an exception will be thrown.

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`, "svec" is a valid sized (as explained above) matrix.
And on exit, If opt_a='N', then the contents of "m" will be destroyed (internally used as workspace).
If opt_a='O' and vtype='L', then "m" will be overwritten with first min(M,N) rows of right singular vectors (stored rowwise in transposed form).
And If opt_a='O' and vtype='R', then "m" will be overwritten with first min(M,N) columns of left singular vectors (stored columnwise).
"sval" is updated with the singular values in sorted oder, so that sval(i) >= sval(i+1) and "svec" will be updated with the desired singular vectors (as explained above).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesvd (m, sval, lsvec, rsvec, part_l='A', part_r='A')

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)
lsvec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)

rsvec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
part_l: A character containing either 'A' or 'S' [Default: 'A'] (input/optional)
part_r: A character containing either 'A' or 'S' [Default: 'A'] (input/optional)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. Additionally, it also computes *left and right singular vectors*.

This function expects that required memory would be allocated for the output matrices "lsvec" and "rsvec" beforehand, to store the left and right singular vectors respectively. If they are not allocated, an exception will be thrown.

If part_l='A', "lsvec" will be updated with all the M columns of left singular vectors. Thus, "lsvec" should have at least M number of rows and M number of columns.
If part_l='S', "lsvec" will be updated with first min(M,N) columns of left singular vectors (stored columnwise). Thus, "lsvec" should have at least M number of rows and min(M,N) number of columns.
If part_r='A', "rsvec" will be updated with all the N rows of right singular vectors (in transposed form). Thus, "rsvec" should have at least N number of rows and N number of columns.
If part_r='S', "rsvec" will be updated with first min(M,N) rows of right singular vectors (stored rowwise in transposed form). Thus, "rsvec" should have at least min(M,N) number of rows and N number of columns.

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`, "lsvec" and "rsvec" are valid sized (as explained above) matrices.
And on exit, the contents of "m" is destroyed (internally used as workspace), "sval" is updated with the singular values in sorted oder, so that sval(i) >= sval(i+1), and "lsvec" and "rvec" are updated with the left and right singular vectors respectively (as explained above).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesdd (m, sval)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. But neither left nor right singular vectors are computed. Please refer to *lapack guide* to know the algorithmic differences between gesvd() and gesdd().

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`. And on successful exit, the contents of "m" is destroyed (used as workspace internally by this function) and "sval" is updated with the singular values in sorted oder, so that sval(i) >= sval(i+1).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesdd (m, sval, svec)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)
svec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. Additionally, it also computes *full or some part of left and right singular vectors* using divide-and-conquer algorithm. Please refer to *lapack documentation* to know the algorithmic differences between gesvd() and gesdd().

If M >= N, matrix "m" will be overwritten with the first N columns of the left singular vectors and "svec" will be updated with all the N rows of right singular vectors (in transposed form). In that case, "svec" should have at least N number of rows and N number of columns.
Otherwise, matrix "m" will be overwritten with first M rows of the right singular vectors (in transposed form) and "svec" will be updated with all the M columns of the left singular vectors. In that case, "svec" should have at least M number of rows and M number of columns.

This function expects that required memory would be allocated for the output matrix "svec" beforehand. If it is not allocated, an exception will be thrown.

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`, "svec" is a valid sized (as explained above) matrix. And on successful exit, "m" and "svec" will be updated with the values (as explained above) and "sval" will be updated with singular values in sorted oder, so that sval(i) >= sval(i+1).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gesdd (m, sval, lsvec, rsvec, part_lr='A')

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty vector of the type `std::vector<T>` (output)
lsvec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
rsvec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
part_lr: A character containing either 'A' or 'S' [Default: 'A'] (input/optional)

Purpose
It computes the singular value decomposition (SVD) of an M-by-N matrix. Additionally, it also computes *full or some part of left and right singular vectors* using divide-and-conquer algorithm. But like the previous version of gesdd (with three parameters), it does not overwrite the results on matrix "m" (since it accepts memory locations for both the left and right singular vectors separately). Please refer to *lapack guide* to know the algorithmic differences between gesvd() and gesdd().

If part_lr='A', all the M columns of left singular vectors and all the N rows of right singular vectors (in transposed form) are stored in output matrix "lsvec" and "rsvec" respectively. In that case "lsvec" should have at least M number of rows and M number of columns and "rsvec" should have at least N number of rows and N number of columns.
If part_lr='S', the first min(M,N) columns of left singular vectors are stored in "lsvec" and the first min(M,N) rows of right singular vectors are stored in "rsvec" (in transposed form). In that case "lsvec" should have at least M number of rows and min(M,N) number of columns and "rsvec" should have at least min(M,N) number of rows and N number of columns.

This function expects that required memory would be allocated for the output matrices "lsvec" and "rsvec" beforehand. If they are not allocated, an exception will be thrown.

On entry "m" contains the matrix whose singular values are to be computed, "sval" is an empty object of the type `std::vector<T>`, "lsvec" and "rsvec" are valid sized (as explained above) matrices. And on successful exit, the contents of "m" will be destroyed (used internally as workspace), "lsvec" and "rsvec" will be updated with the values (as explained above) and "sval" will be updated with singular values in sorted oder, so that sval(i) >= sval(i+1).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelsy (m1, m2, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
It computes the minimum-norm solution to a real linear least squares problem:

```
minimize || A * X - B ||
```

using a complete orthogonal factorization of A. A is an M-by-N matrix which may be rank-deficient.

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, "m1" is overwritten with its complete orthogonal factorization and "m2" is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelsy (m1, m2, rank, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
rank: An empty integer object passed by reference (output)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
The native lapack routine can also determine the rank of the matrix "m1" while finding the minimum-norm solution. If it is required to know the rank determined by this function, it is recommended to use this version of gelsy().

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B), "rank" is just an empty integer passed by reference to this routine. On successful exit, "m1" is overwritten with its complete orthogonal factorization, "m2" is overwritten with the solution matrix (X) and "rank" is overwritten with the determined effective rank of matrix "m1".

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelss (m1, m2, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
It solves overdetermined or underdetermined systems for general matrices. It computes the minimum-norm solution to a real linear least squares problem:

```
minimize 2-norm (| B - AX |)
```

using the singular value decomposition (SVD) of A. A is an M-by-N general matrix which may be rank-deficient.

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, first min(M,N) rows of "m1" is overwritten with its right singular vectors (stored rowwise) and "m2" is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelss (m1, m2, sval, rank, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty object of the type `std::vector<T>` (output)
rank: An empty integer object passed by reference (output)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
It solves overdetermined or underdetermined systems for general matrices. It computes the minimum-norm solution to a real linear least squares problem:

```
minimize 2-norm (| B - AX |)
```

using the singular value decomposition of A. A is an M-by-N general matrix which may be rank-deficient.

It might also be needed to obtain the computed singular values and effective rank of the matrix A. In that case, this version of gelss(with five arguments) is recommended to use. It accepts an empty vector (3rd argument) and an empty integer (4th argument) which are passed by reference to this function.

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, first min(M,N) rows of "m1" is overwritten with its right singular vectors (stored rowwise), "m2" is overwritten with the solution matrix (X), computed singular values of of "m1" are stored in "sval" in decreasing order and "rank" is updated with the computed effective rank of "m1".

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelsd (m1, m2, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
It solves overdetermined or underdetermined systems for general matrices. It computes the minimum-norm solution to a real linear least squares problem:

```
minimize 2-norm (| B - AX |)
```

using the singular value decomposition (SVD) of A. A is an M-by-N general matrix which may be rank-deficient. Please refer to *lapack guide* to know the algorithmic differences between gelsd() and gelss().

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, first min(M,N) rows of "m1" is overwritten with its right singular vectors (stored rowwise) and "m2" is overwritten with the solution matrix (X).

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

gelsd (m1, m2, sval, rank, rcond=-1)

Parameters
m1: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
m2: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
sval: An empty object of the type `std::vector<T>` (output)
rank: An empty integer object passed by reference (output)
rcond: A T type object (float or double) [Default: -1] (input/optional)

Purpose
It solves overdetermined or underdetermined systems for general matrices. It computes the minimum-norm solution to a real linear least squares problem:

```
minimize 2-norm (| B - AX |)
```

using the singular value decomposition of A. A is an M-by-N general matrix which may be rank-deficient. Please refer to *lapack guide* to know the algorithmic differences between gelsd() and gelss().

It might also be needed to obtain the computed singular values and effective rank of the matrix A. In that case, this version of gelsd(with five arguments) is recommended to use. It accepts an empty vector (3rd argument) and an empty integer (4th argument) which are passed by reference to this function.

The input parameter "rcond" is used to determine the effective rank of matrix "m1". If "rcond" is less than zero, machine precision is used instead. The matrix "m2" should have number of rows >= max(M,N) and at least 1 column.

On entry, "m1" contains the left-hand-side matrix (A) and "m2" contains the right-hand-side matrix (B). On successful exit, first min(M,N) rows of "m1" is overwritten with its right singular vectors (stored rowwise), "m2" is overwritten with the solution matrix (X), computed singular values of of "m1" are stored in "sval" in decreasing order and "rank" is updated with the computed effective rank of "m1".

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

geev (m, eval)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
eval: An empty object of the type `std::vector<T>` (output)

Purpose
It computes eigenvalues for an N-by-N real nonsymmetric matrix.

The input matrix, "m" must be a square matrix. Else it will throw an exception.

On entry, "m" is the square matrix whose eigenvalues are to be computed and "eval" is an empty vector. On successful exit, the contents of "m" is destroyed, and the computed eigenvalues are stored in "eval".

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

geev (m, eval, evec, vtype='L')

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
eval: An empty object of the type `std::vector<T>` (output)
evec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
vtype: A character value containing either 'L' or 'R' [Default: 'L'] (input/optional)

Purpose
It computes eigenvalues for an N-by-N real nonsymmetric matrix. Additionally, it also computes the left or right eigenvectors.

The input matrix, "m" must be a square matrix. Else it will throw an exception. If vtype='L', then left-eigenvectors will be computed.
If vtype='R', then right-eigenvectors will be computed. The output matrix "evec" must have at least N number of rows and N number of columns. This function expects that "evec" is already allocated before its call. Thus if it is not allocated, an exception will be thrown.

On entry, "m" is the square matrix whose eigenvalues are to be computed, "eval" is an empty vector, "evec" is an empty matrix with valid size (as mentioned above). On successful exit, the contents of "m" is destroyed, the computed eigenvalues are stored in "eval" and "evec" is updated with the desired (left/right) eigenvectors.

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

geev (m, eval, levec, revec)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (inout)
eval: An empty object of the type `std::vector<T>` (output)
levec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)
revec: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (output)

Purpose
It computes eigenvalues for an N-by-N real nonsymmetric matrix. Additionally, it also computes the left and right eigenvectors.

The input matrix, "m" must be a square matrix. Else it will throw an exception. The output matrices "levec" and "revec" must have at least N number of rows and N number of columns. This function expects that these output matrices are already allocated before its call. Thus if they are not allocated, an exception will be thrown.

On entry, "m" is the square matrix whose eigenvalues are to be computed, "eval" is an empty vector, "levec" and "revec" are an empty matrices with valid size (as mentioned above). On successful exit, the contents of "m" is destroyed, the computed eigenvalues are stored in "eval", "levec" is updated with the left eigenvectors and "revec" is updated with right eigenvectors.

Return Value
On success, it returns the exit status of the lapack routine itself. If any error occurs, it throws an exception explaining cause of the error.

inv (m)

Parameters
m: A `colmajor_matrix_local<T>` or a `sliced_colmajor_matrix_local<T>` (input)

Purpose
It computes the inverse of a square matrix "m" by using getrf() and getri() internally. Thus it is a kind of short-cut function to obtain the inverse of a non-distributed matrix.

On successful exit, it returns the resultant inversed matrix. The input matrix "m" remains unchanged. Since it returns the resultant matrix, it can be used in any numerical expressions, along with other operators. E.g., if a and b are two colmajor matrices, then the expresion like, "a*(~b)*inv(a)" can easily be performed.

Return Value
On success, it returns the resultant matrix of the type `colmajor_matrix_local<T>`. If any error occurs, it throws an exception explaining cause of the error.

SEE ALSO

sliced_colmajor_matrix_local, sliced_colmajor_vector_local, scalapack_wrapper