hipSOLVER Documentation

Release 2.2.0

Advanced Micro Devices, Inc.

Nov 07, 2024

INSTALL

hipSOLVER is a LAPACK marshalling library, with multiple supported backends. It sits between the application and a 'worker' LAPACK library, marshalling inputs into the backend library and marshalling results back to the application. hipSOLVER supports rocSOLVER and cuSOLVER as backends. hipSOLVER exports an interface that does not require the client to change, regardless of the chosen backend.

The code is open and hosted at: https://github.com/ROCm/hipSOLVER

The hipSOLVER documentation is structured as follows:

Installation

- *Installation on Linux*

How to

- *Using hipSOLVER*

Reference

- *Introduction to hipSOLVER API*
- *hipSOLVER regular API*
- *hipSOLVER compatibility API - Dense Matrices*
- *hipSOLVER compatibility API - Sparse Matrices*
- *hipSOLVER Compatibility API - Refactorization*

Using hipSOLVER is the starting point for new users of the library. For a list of currently implemented routines in the different APIs refer to *Introduction to hipSOLVER API*.

To contribute to the documentation refer to Contributing to ROCm.

You can find licensing information on the Licensing page.

CHAPTER

ONE

INSTALLATION ON LINUX

1.1 Install pre-built packages

Download pre-built packages from ROCm's package servers. Updates to each release are listed in the `CHANGELOG.md` file under the releases tab of the hipSOLVER GitHub page.

- `sudo apt update && sudo apt install hipsolver`

> **Note**
>
> The pre-built packages depend on the third-party library SuiteSparse, which must be installed on the system prior to installing hipSOLVER. SuiteSparse can be installed using the package manager of most distros.

1.2 Build & install library using script (Ubuntu only)

The root of the hipSOLVER repository has a helper bash script `install.sh` to build and install hipSOLVER on Ubuntu with a single command. It does not take a lot of options and hard-codes configuration that can be specified through invoking `cmake` directly, but it's a great way to get started quickly and can serve as an example of how to build/install. A few commands in the script need sudo access, so it may prompt you for a password.

- `./install.sh -id` — build library, build dependencies, and install (-d flag only needs to be passed once on a system).
- `./install.sh -ic` — build library, build clients (tests, benchmarks, and samples), and install.
- `./install.sh --cuda` — build library on a CUDA-enabled machine, with cuSOLVER as the backend.
- `./install.sh --no-sparse` — build library without hipsolverSp functionality, with rocSOLVER as the backend.

To see more options, use the help option of the install script.

- `./install.sh -h`

1.3 Build & install library manually

For a standard library installation, follow these steps:

```
mkdir -p <HIPSOLVER_BUILD_DIR_PATH>/release
cd <HIPSOLVER_BUILD_DIR_PATH>/release
CXX=/opt/rocm/bin/hipcc cmake <HIPSOLVER_SOURCE_DIR_PATH>
```

(continues on next page)

(continued from previous page)

```
make -j$(nproc)
sudo make install
```

sudo is required if installing into a system directory such as /opt/rocm, which is the default option.

- Use `-DCMAKE_INSTALL_PREFIX=<other_path>` to specify a different install directory.
- Use `-DCMAKE_BUILD_TYPE=<other_configuration>` to specify a build configuration, such as 'Debug'. The default build configuration is 'Release'.

1.3.1 Library dependencies

The hipSOLVER library has two separate sets of dependencies, depending on the desired backend. The cuSOLVER backend has the following dependencies:

1. cuSOLVER

The rocSOLVER backend has the following dependencies:

1. rocSOLVER
2. rocBLAS
3. rocSPARSE (optional, required by default)
4. SuiteSparse modules CHOLMOD and SuiteSparse_config (optional, required by default)

rocSOLVER itself depends on rocBLAS and rocSPARSE, therefore all three libraries should be present with a standard rocSOLVER installation. For more information about building and installing rocSOLVER, refer to the rocSOLVER documentation.

SuiteSparse is a third-party library, and can be installed using the package managers of most distros. Together with rocSPARSE, it is used to provide functionality for the hipsolverSp API. If only hipsolverDn and/or hipsolverRf are needed, these dependencies can be ignored by setting the `BUILD_WITH_SPARSE` option to `OFF`.

- `DBUILD_WITH_SPARSE=OFF`

1.4 Build library + tests + benchmarks + samples manually

The repository contains source code for client programs that serve as tests, benchmarks, and samples. Client source code can be found in the clients subdirectory.

1.4.1 Client dependencies

The hipSOLVER samples have no external dependencies, but our unit test and benchmarking applications do. These clients introduce the following dependencies:

1. lapack (lapack itself brings a dependency on a fortran compiler)
2. googletest
3. hipBLAS (optional)
4. hipSPARSE (optional, required by default)

Unfortunately, many distros do not provide a googletest package with pre-compiled libraries, and the lapack packages do not have the necessary cmake config files for cmake to configure linking the cblas library. hipSOLVER provides a cmake script that builds lapack and googletest from source. This is an optional step; users can provide their own builds of these dependencies and help cmake find them by setting the `CMAKE_PREFIX_PATH` definition. The following is a sequence of steps to build dependencies and install them to the cmake default, /usr/local:

```
mkdir -p <HIPSOLVER_BUILD_DIR_PATH>/release/deps
cd <HIPSOLVER_BUILD_DIR_PATH>/release/deps
cmake -DBUILD_BOOST=OFF <HIPSOLVER_SOURCE_PATH>/deps   #assuming boost is installed␣
↪through package manager as above
make -j$(nproc) install
```

hipBLAS is only required if the BUILD_HIPBLAS_TESTS option is set to ON, and is used to ensure compatibility between the hipblas enums defined separately by hipBLAS and hipSOLVER. hipSPARSE is required by default but can be ignored if the BUILD_WITH_SPARSE option is set to OFF, and is used to create objects required by tests for the hipsolverSp API.

- DBUILD_HIPBLAS_TESTS=ON
- DBUILD_WITH_SPARSE=OFF

Both libraries can be installed similarly to hipSOLVER. For example, the install scripts for hipBLAS and hipSPARSE can each be invoked to build and install the respective library via:

- ./install.sh -i

Find more details in the hipBLAS documentation and the hipSPARSE documentation.

1.4.2 Library and clients

Once dependencies are available on the system, it is possible to configure the clients to build. This requires a few extra cmake flags to the library's cmake configure script. If the dependencies are not installed into system defaults (like /usr/local), you should pass the CMAKE_PREFIX_PATH to cmake to help find them.

- -DCMAKE_PREFIX_PATH="<semicolon separated paths>"

```
CXX=/opt/rocm/bin/hipcc cmake -DBUILD_CLIENTS_TESTS=ON -DBUILD_CLIENTS_BENCHMARKS=ON␣
↪[HIPSOLVER_SOURCE]
make -j$(nproc)
sudo make install   # sudo required if installing into system directory such as /opt/rocm
```

CHAPTER

TWO

USING HIPSOLVER

hipSOLVER is an open-source marshalling library for LAPACK routines on the GPU. It sits between a backend library and the user application, marshalling inputs to and outputs from the backend library so that the user application remains unchanged when using different backends. Currently, two backend libraries are supported by hipSOLVER: NVIDIA's cuSOLVER library and AMD's open-source rocSOLVER library.

The *regular hipSOLVER API* is a thin wrapper layer around the different backends. As such, it is not expected to introduce significant overhead. However, its main purpose is portability, so when performance is critical directly using the library backend is recommended.

Once installed, hipSOLVER can be used just like any other library with a C API. The header file will need to be included in the user code, and the shared library will become link-time and run-time dependencies for the user application. The user code can be ported, with no changes, to any system with hipSOLVER installed regardless of the backend library.

For more details on how to use the API methods, see the code samples on hipSOLVER's clients' GitHub page, or the documentation of the corresponding backend libraries.

2.1 Porting cuSOLVER applications to hipSOLVER

Another purpose of hipSOLVER is to facilitate the translation of cuSOLVER applications to AMD's open-source ROCm platform ecosystem. hipSOLVER is designed to make it easy for users of cuSOLVER to port their existing applications to hipSOLVER, and provides two separate but interchangeable API patterns in order to facilitate a two-stage transition process. Users are encouraged to start with hipSOLVER's compatibility APIs, which use the *hipsolverDn*, *hipsolverSp*, and *hipsolverRf* prefixes and have method signatures that are fully consistent with cuSOLVER functions.

However, the compatibility APIs may introduce some performance drawbacks, especially when using the rocSOLVER backend. So, as a second stage, it is recommended to begin the switch to hipSOLVER's *regular API* when possible. The regular API uses the *hipsolver* prefix and introduces minor adjustments to the API in order to get the best performance out of the rocSOLVER backend. In most cases, switching to the regular API is as simple as removing *Dn* from the *hipsolverDn* prefix (methods with the *hipsolverSp* and *hipsolverRf* prefixes are not currently supported by the regular API).

No matter which API is used, a hipSOLVER application can be executed, without modifications to the code, in systems with cuSOLVER or rocSOLVER installed. However, using the regular API ensures the best performance out of both backends.

2.2 Some considerations when using the hipsolverDn API

The hipsolverDn API is intended as a 1:1 translation of the cusolverDn API, but not all functionality is equally supported in rocSOLVER. Keep in mind the following considerations when using this compatibility API.

2.2.1 Arguments not referenced by rocSOLVER

- Unlike cuSOLVER, rocSOLVER functions do not provide information on invalid arguments in the *info* parameter, though they will provide info on singularities and algorithm convergence. Hence, when using the rocSOLVER backend, *info* will always return a value >= 0. In those cases where a rocSOLVER function does not accept *info* as an argument, hipSOLVER will set it to zero.
- The *niters* argument of *hipsolverDnXXgels* and *hipsolverDnXXgesv* is not referenced by the rocSOLVER backend; there is no iterative refinement currently implemented in rocSOLVER.
- The *hRnrmF* argument of *hipsolverDnXgesvdaStridedBatched* is not referenced by the rocSOLVER backend.

2.2.2 Performance implications of the hipsolverDn API

- To calculate the workspace required by function *gesvd* in rocSOLVER, the values of *jobu* and *jobv* are needed, however, the function *hipsolverDnXgesvd_bufferSize* does not accept these arguments. So, when using the rocSOLVER backend, *hipsolverDnXgesvd_bufferSize* has to calculate internally the workspace for all possible values of *jobu* and *jobv*, and return the maximum.

  (*hipsolverDnXgesvd_bufferSize* is slower than *hipsolverXgesvd_bufferSize*, and its returned workspace size could be slightly larger than what is actually needed).
- To properly use a user-provided workspace, rocSOLVER requires both the allocated pointer and its size. However, the function *hipsolverDnXgetrf* does not accept *lwork* as an argument. In consequence, when using the rocSOLVER backend, *hipsolverDnXgetrf* has to call internally *hipsolverDnXgetrf_bufferSize* to know the size of the workspace.

  (*hipsolverDnXgetrf_bufferSize* will be called twice in practice, once by the user before allocating the workspace, and once by hipSOLVER internally when executing the *hipsolverDnXgetrf* function. *hipsolverDnXgetrf* could be slightly slower than *hipsolverXgetrf* because of the extra call to the bufferSize helper).
- The functions *hipsolverDnXgetrs*, *hipsolverDnXpotrs*, *hipsolverDnXpotrsBatched*, and *hipsolverDnXpotrfBatched* do not accept *work* and *lwork* as arguments. However, this functionality does require a non-zero workspace in rocSOLVER. As a result, when using the rocSOLVER backend, these functions will switch to the automatic workspace management model (see *here*).

  (Users must keep in mind that even if the compatibility API does not have bufferSize helpers for the mentioned functions, these functions do require workspace when using rocSOLVER, and it will be automatically managed. This may imply device memory reallocations with corresponding overheads).

2.3 Some considerations when using the hipsolverSp API

The hipsolverSp API is intended as a 1:1 translation of the cusolverSp API, but not all functionality is equally supported in rocSOLVER. Keep in mind the following considerations when using this compatibility API.

2.3.1 Unsupported methods

- RCM reordering is currently not supported by rocSOLVER, rocSPARSE, and SuiteSparse. The following methods will instead use AMD reordering when RCM is requested.
  - *hipsolverSpXcsrlsvcholHost* with *reorder = 1*
  - *hipsolverSpXcsrlsvchol* with *reorder = 1*

2.3.2 Performance implications of the hipsolverSp API

- The third-party SuiteSparse library is used to provide host-side functionality for the hipsolverSp API when using the rocSOLVER backend. At present, SuiteSparse does not support single precision arrays, therefore hipSOLVER must allocate temporary double precision arrays and copy the values one-by-one to and from the user-provided arguments.

  (Single precision hipsolverSp functions are expected to perform slower and require more memory usage than double precision functions.)

- A fully-featured, GPU-accelerated Cholesky factorization for sparse matrices has not yet been implemented in either rocSOLVER or rocSPARSE. Therefore, we rely on SuiteSparse to provide this functionality. The functions *hipsolverSpXcsrlsvchol* will allocate space for sparse matrices on the host, copy the data to the host, use SuiteSparse to perform the symbolic factorization, and then copy the resulting data back to the device.

  (*hipsolverSpXcsrlsvchol* may perform slower and will require more memory usage than *hipsolverSpXcsrlsvcholHost*.)

2.4 Some considerations when using the hipsolverRf API

The hipsolverRf API is intended as a 1:1 translation of the cusolverRf API, but not all functionality is equally supported in rocSOLVER. Keep in mind the following considerations when using this compatibility API.

2.4.1 Unsupported methods

- Batched refactorization methods are currently unsupported with the rocSOLVER backend and will return a *HIPSOLVER_STATUS_NOT_SUPPORTED* status code.
  - *hipsolverRfBatchSetupHost*
  - *hipsolverRfBatchAnalyze*
  - *hipsolverRfBatchResetValues*
  - *hipsolverRfBatchZeroPivot*
  - *hipsolverRfBatchRefactor*
  - *hipsolverRfBatchSolve*
- Parameter setting methods are currently unsupported with the rocSOLVER backend and will return a *HIPSOLVER_STATUS_NOT_SUPPORTED* status code.
  - *hipsolverRfSetAlgs*
  - *hipsolverRfSetMatrixFormat*
  - *hipsolverRfSetNumericProperties*
  - *hipsolverRfSetResetValuesFastMode*

2.5 Some considerations when using the regular hipSOLVER API

hipSOLVER's regular API is similar to cuSOLVER; however, due to differences in the implementation and design between cuSOLVER and rocSOLVER, some minor adjustments were introduced to ensure the best performance out of both backends.

2.5.1 Different signatures and additional API methods

- The methods to obtain the size of the workspace needed by functions *gels* and *gesv* in cuSOLVER require *dwork* as an argument; however, it is never used and can be null. On the rocSOLVER side, *dwork* is not needed to calculate the workspace size. In consequence:
  - *hipsolverXXgels_bufferSize* does not require *dwork* as an argument, and
  - *hipsolverXXgesv_bufferSize* does not require *dwork* as an argument.

  (These wrappers pass *dwork* = *nullptr* when calling cuSOLVER).
- To calculate the workspace required by function *gesvd* in rocSOLVER, the values of *jobu* and *jobv* are needed. As a result,
  - *hipsolverXgesvd_bufferSize* requires *jobu* and *jobv* as arguments.

  (These arguments are ignored when the wrapper calls cuSOLVER, as they are not needed).
- To properly use a user-provided workspace, rocSOLVER requires both the allocated pointer and its size. Consequently:
  - *hipsolverXgetrf* requires *lwork* as an argument.

  (*lwork* is ignored when the wrapper calls cuSOLVER, as it is not needed).
- All rocSOLVER functions called by hipSOLVER require a workspace. To allow the user to specify one,
  - *hipsolverXgetrs* requires *work* and *lwork* as arguments,
  - *hipsolverXpotrfBatched* requires *work* and *lwork* as arguments,
  - *hipsolverXpotrs* requires *work* and *lwork* as arguments, and
  - *hipsolverXpotrsBatched* requires *work* and *lwork* as arguments.

  (These arguments are ignored when these wrappers call cuSOLVER, as they are not needed).

  In order to support these changes, the regular API adds the following functions as well:
  - *hipsolverXgetrs_bufferSize*
  - *hipsolverXpotrfBatched_bufferSize*
  - *hipsolverXpotrs_bufferSize*
  - *hipsolverXpotrsBatched_bufferSize*

  (These methods return *lwork* = *0* when using the cuSOLVER backend, as the corresponding functions in cuSOLVER do not need workspace).

2.5.2 Arguments not referenced by rocSOLVER

- Unlike cuSOLVER, rocSOLVER functions do not provide information on invalid arguments in the *info* parameter, though they will provide info on singularities and algorithm convergence. Hence, when using the rocSOLVER backend, *info* will always return a value >= 0. In those cases where a rocSOLVER function does not accept *info* as an argument, hipSOLVER will set it to zero.
- The *niters* argument of *hipsolverXXgels* and *hipsolverXXgesv* is not referenced by the rocSOLVER backend; there is no iterative refinement currently implemented in rocSOLVER.

2.5.3 Using rocSOLVER's memory model

Most hipSOLVER functions take a workspace pointer and size as arguments, allowing the user to manage the device memory used internally by the backends. rocSOLVER, however, can maintain the device workspace automatically by default (see rocSOLVER's memory model for more details). In order to take advantage of this feature, users may pass a null pointer for the *work* argument or a zero size for the *lwork* argument of any function when using the rocSOLVER backend, and the workspace will be automatically managed behind-the-scenes. It is recommended, however, to use a consistent strategy for workspace management, as performance issues may arise if the internal workspace is made to flip-flop between user-provided and automatically allocated workspaces.

Warning

This feature should not be used with the cuSOLVER backend; hipSOLVER does not guarantee a defined behavior when passing a null workspace to cuSOLVER functions that require one.

2.5.4 Using rocSOLVER's in-place functions

The solvers *gesv* and *gels* in cuSOLVER are out-of-place in the sense that the solution vectors X do not overwrite the input matrix B. In rocSOLVER this is not the case; when *hipsolverXXgels* or *hipsolverXXgesv* call rocSOLVER, some data movements must be done internally to restore B and copy the results back to X. These copies could introduce noticeable overhead depending on the size of the matrices. To avoid this potential problem, users can pass $X = B$ to *hipsolverXXgels* or *hipsolverXXgesv* when using the rocSOLVER backend; in this case, no data movements will be required, and the solution vectors can be retrieved using either B or X.

Warning

This feature should not be used with the cuSOLVER backend; hipSOLVER does not guarantee a defined behavior when passing $X = B$ to the mentioned functions in cuSOLVER.

CHAPTER

THREE

INTRODUCTION TO HIPSOLVER API

Note

The hipSOLVER library remains in active development. New features are being continuously added, with new functionality documented at each release of the ROCm platform.

The following tables summarize the wrapper functions that are implemented in the regular API for the different supported precisions in latest hipSOLVER release. Most of these functions have a corresponding version in the compatibility APIs, where applicable.

3.1 LAPACK auxiliary functions

Table 3.1: Orthonormal matrices

Function	single	double	single complex	double complex
hipsolverXorgbr_bufferSize	x	x		
hipsolverXorgbr	x	x		
hipsolverXorgqr_bufferSize	x	x		
hipsolverXorgqr	x	x		
hipsolverXorgtr_bufferSize	x	x		
hipsolverXorgtr	x	x		
hipsolverXormqr_bufferSize	x	x		
hipsolverXormqr	x	x		
hipsolverXormtr_bufferSize	x	x		
hipsolverXormtr	x	x		

Table 3.2: Unitary matrices

Function	single	double	single complex	double complex
hipsolverXungbr_bufferSize			x	x
hipsolverXungbr			x	x
hipsolverXungqr_bufferSize			x	x
hipsolverXungqr			x	x
hipsolverXungtr_bufferSize			x	x
hipsolverXungtr			x	x
hipsolverXunmqr_bufferSize			x	x
hipsolverXunmqr			x	x
hipsolverXunmtr_bufferSize			x	x
hipsolverXunmtr			x	x

3.2 LAPACK main functions

Table 3.3: Triangular factorizations

Function	single	double	single complex	double complex
hipsolverXpotrf_bufferSize	x	x	x	x
hipsolverXpotrf	x	x	x	x
hipsolverXpotrfBatched_bufferSize	x	x	x	x
hipsolverXpotrfBatched	x	x	x	x
hipsolverXgetrf_bufferSize	x	x	x	x
hipsolverXgetrf	x	x	x	x
hipsolverXsytrf_bufferSize	x	x	x	x
hipsolverXsytrf	x	x	x	x

Table 3.4: Orthogonal factorizations

Function	single	double	single complex	double complex
hipsolverXgeqrf_bufferSize	x	x	x	x
hipsolverXgeqrf	x	x	x	x

Table 3.5: Problem and matrix reductions

Function	single	double	single complex	double complex
hipsolverXsytrd_bufferSize	x	x		
hipsolverXsytrd	x	x		
hipsolverXhetrd_bufferSize			x	x
hipsolverXhetrd			x	x
hipsolverXgebrd_bufferSize	x	x	x	x
hipsolverXgebrd	x	x	x	x

Table 3.6: Linear-systems solvers

Function	single	double	single complex	double complex
hipsolverXpotri_bufferSize	x	x	x	x
hipsolverXpotri	x	x	x	x
hipsolverXpotrs_bufferSize	x	x	x	x
hipsolverXpotrs	x	x	x	x
hipsolverXpotrsBatched_bufferSize	x	x	x	x
hipsolverXpotrsBatched	x	x	x	x
hipsolverXgetrs_bufferSize	x	x	x	x
hipsolverXgetrs	x	x	x	x
hipsolverXXgesv_bufferSize	x	x	x	x
hipsolverXXgesv	x	x	x	x

Table 3.7: Least-square solvers

Function	single	double	single complex	double complex
hipsolverXXgels_bufferSize	x	x	x	x
hipsolverXXgels	x	x	x	x

Table 3.8: Symmetric eigensolvers

Function	single	double	single complex	double complex
hipsolverXsyevd_bufferSize	x	x		
hipsolverXsyevd	x	x		
hipsolverXsygvd_bufferSize	x	x		
hipsolverXsygvd	x	x		
hipsolverXheevd_bufferSize			x	x
hipsolverXheevd			x	x
hipsolverXhegvd_bufferSize			x	x
hipsolverXhegvd			x	x

Table 3.9: Singular value decomposition

Function	single	double	single complex	double complex
hipsolverXgesvd_bufferSize	x	x	x	x
hipsolverXgesvd	x	x	x	x

3.3 LAPACK-like functions

Table 3.10: Symmetric eigensolvers

Function	single	double	single complex	double complex
hipsolverXsyevdx_bufferSize	x	x		
hipsolverXsyevdx	x	x		
hipsolverXsyevj_bufferSize	x	x		
hipsolverXsyevj	x	x		
hipsolverXsyevjBatched_bufferSize	x	x		
hipsolverXsyevjBatched	x	x		
hipsolverXsygvdx_bufferSize	x	x		
hipsolverXsygvdx	x	x		
hipsolverXsygvj_bufferSize	x	x		
hipsolverXsygvj	x	x		
hipsolverXheevdx_bufferSize			x	x
hipsolverXheevdx			x	x
hipsolverXheevj_bufferSize			x	x
hipsolverXheevj			x	x
hipsolverXheevjBatched_bufferSize			x	x
hipsolverXheevjBatched			x	x
hipsolverXhegvdx_bufferSize			x	x
hipsolverXhegvdx			x	x
hipsolverXhegvj_bufferSize			x	x
hipsolverXhegvj			x	x

Table 3.11: Singular value decomposition

Function	single	double	single complex	double complex
hipsolverDnXgesvdj_bufferSize	x	x	x	x
hipsolverDnXgesvdj	x	x	x	x
hipsolverDnXgesvdjBatched_bufferSize	x	x	x	x
hipsolverDnXgesvdjBatched	x	x	x	x

3.3.1 Compatibility-only functions

The following tables summarize the wrapper functions that are provided only in the compatibility APIs. These wrappers are supported in rocSOLVER but either by equivalent functions that use different algorithmic approaches, or by functionality that is not fully exposed in the public API. For these reasons, at present, the corresponding wrappers are not provided in the regular hipSOLVER API.

3.4 Partial SVD functions

Partial SVD has been implemented in rocSOLVER, but at present it does not use an approximate algorithm, nor does it compute the residual norm.

Table 3.12: Singular value decomposition

Function	single	double	single complex	double complex
hipsolverDnXgesvdaStridedBatched_bufferSize	x	x	x	x
hipsolverDnXgesvdaStridedBatched	x	x	x	x

3.5 Sparse matrix routines

Sparse matrix routines and direct solvers for sparse matrices are in the very earliest stages of development. Due to unsupported backend functionality, there are a number of intricacies and possible performance implications that users will want to be aware of when using these routines. Refer to the *hipsolverSp compatibility API* for more details and a full listing of supported functions.

Table 3.13: Combined factorization and linear-system solvers

Function	single	double	single complex	double complex
hipsolverSpXcsrlsvcholHost	x	x		
hipsolverSpXcsrlsvchol	x	x		

3.6 Refactorization routines

Refactorization routines and direct solvers for sparse matrices are in the very earliest stages of development. Due to unsupported backend functionality, there are a number of intricacies and possible performance implications that users will want to be aware of when using these routines. Refer to the *hipsolverRf compatibility API* for more details and a full listing of supported functions.

Table 3.14: Triangular factorizations

Function	single	double	single complex	double complex
hipsolverRfRefactor	x	x		
hipsolverRfBatchRefactor	x	x		

Table 3.15: linear-system solvers

Function	single	double	single complex	double complex
hipsolverRfSolve	x	x		
hipsolverRfBatchSolve	x	x		

CHAPTER

FOUR

HIPSOLVER REGULAR API

This document provides the method signatures for wrapper functions that are currently implemented in hipSOLVER. For a complete description of the functions' behavior and arguments, see the corresponding backend documentation at cuSOLVER API and/or rocSOLVER API.

The hipSOLVER API is designed to be similar to the cuSOLVER and rocSOLVER interfaces, but it requires some minor adjustments to ensure the best performance out of both backends. Generally, this involves the addition of workspace parameters and some additional API methods. Refer to *Using hipSOLVER* for a complete list of *API differences*.

Users interested in using hipSOLVER without these adjustments, so that the interface matches cuSOLVER, should instead consult the *Compatibility API documentation*. See also *the porting section* for more details.

- *hipSOLVER datatypes*
- *hipSOLVER helper functions*
- *hipSOLVER LAPACK auxiliary functions*
- *hipSOLVER LAPACK Functions*
- *hipSOLVER LAPACK-like functions*

4.1 hipSOLVER datatypes

hipSOLVER defines types and enumerations that are internally converted to the corresponding backend types at runtime. Here we list the types used in the regular API.

4.1.1 hipSOLVER regular API types

hipsolverHandle_t

typedef void *`hipsolverHandle_t`

hipsolverGesvdjInfo_t

typedef void *`hipsolverGesvdjInfo_t`

hipsolverSyevjInfo_t

typedef void *`hipsolverSyevjInfo_t`

hipsolverStatus_t

enum **hipsolverStatus_t**

Values:

enumerator **HIPSOLVER_STATUS_SUCCESS**

enumerator **HIPSOLVER_STATUS_NOT_INITIALIZED**

enumerator **HIPSOLVER_STATUS_ALLOC_FAILED**

enumerator **HIPSOLVER_STATUS_INVALID_VALUE**

enumerator **HIPSOLVER_STATUS_MAPPING_ERROR**

enumerator **HIPSOLVER_STATUS_EXECUTION_FAILED**

enumerator **HIPSOLVER_STATUS_INTERNAL_ERROR**

enumerator **HIPSOLVER_STATUS_NOT_SUPPORTED**

enumerator **HIPSOLVER_STATUS_ARCH_MISMATCH**

enumerator **HIPSOLVER_STATUS_HANDLE_IS_NULLPTR**

enumerator **HIPSOLVER_STATUS_INVALID_ENUM**

enumerator **HIPSOLVER_STATUS_UNKNOWN**

enumerator **HIPSOLVER_STATUS_ZERO_PIVOT**

enumerator **HIPSOLVER_STATUS_MATRIX_TYPE_NOT_SUPPORTED**

hipblasOperation_t

enum **hipblasOperation_t**

Used to specify whether the matrix is to be transposed or not.

Values:

enumerator **HIPBLAS_OP_N**

Operate with the matrix.

enumerator **HIPBLAS_OP_T**

Operate with the transpose of the matrix.

enumerator **HIPBLAS_OP_C**

Operate with the conjugate transpose of the matrix.

hipsolverOperation_t

typedef *hipblasOperation_t* **hipsolverOperation_t**

Alias of hipblasOperation_t. HIPSOLVER_OP_N, HIPSOLVER_OP_T, and HIPSOLVER_OP_C are provided as equivalents to HIPBLAS_OP_N, HIPBLAS_OP_T, and HIPBLAS_OP_C.

hipblasFillMode_t

enum **hipblasFillMode_t**

Used by the Hermitian, symmetric and triangular matrix routines to specify whether the upper or lower triangle is being referenced.

Values:

enumerator **HIPBLAS_FILL_MODE_UPPER**

Upper triangle

enumerator **HIPBLAS_FILL_MODE_LOWER**

Lower triangle

enumerator **HIPBLAS_FILL_MODE_FULL**

hipsolverFillMode_t

typedef *hipblasFillMode_t* **hipsolverFillMode_t**

Alias of hipblasFillMode_t. HIPSOLVER_FILL_MODE_UPPER and HIPSOLVER_FILL_MODE_LOWER are provided as equivalents to HIPBLAS_FILL_MODE_UPPER and HIPBLAS_FILL_MODE_LOWER.

hipblasSideMode_t

enum **hipblasSideMode_t**

Indicates the side matrix A is located relative to matrix B during multiplication.

Values:

enumerator **HIPBLAS_SIDE_LEFT**

Multiply general matrix by symmetric, Hermitian or triangular matrix on the left.

enumerator **HIPBLAS_SIDE_RIGHT**

Multiply general matrix by symmetric, Hermitian or triangular matrix on the right.

enumerator **HIPBLAS_SIDE_BOTH**

hipsolverSideMode_t

typedef *hipblasSideMode_t* **hipsolverSideMode_t**

Alias of hipblasSideMode_t. HIPSOLVER_SIDE_LEFT and HIPSOLVER_SIDE_RIGHT are provided as equivalents to HIPBLAS_SIDE_LEFT and HIPBLAS_SIDE_RIGHT.

hipsolverEigMode_t

enum **hipsolverEigMode_t**

Values:

enumerator **HIPSOLVER_EIG_MODE_NOVECTOR**

enumerator **HIPSOLVER_EIG_MODE_VECTOR**

hipsolverEigType_t

enum **hipsolverEigType_t**

Values:

enumerator **HIPSOLVER_EIG_TYPE_1**

enumerator **HIPSOLVER_EIG_TYPE_2**

enumerator **HIPSOLVER_EIG_TYPE_3**

hipsolverEigRange_t

enum **hipsolverEigRange_t**

Values:

enumerator **HIPSOLVER_EIG_RANGE_ALL**

enumerator **HIPSOLVER_EIG_RANGE_V**

enumerator **HIPSOLVER_EIG_RANGE_I**

4.2 hipSOLVER helper functions

These are helper functions that control aspects of the hipSOLVER library. They are divided into the following categories:

- *Handle set-up and tear-down* functions. Used to initialize and cleanup the library handle.
- *Stream manipulation* functions. Provide functionality to manipulate streams.
- *Gesvdj parameter manipulation* functions. Provide functionality to manipulate gesvdj parameters.
- *Syevj parameter manipulation* functions. Provide functionality to manipulate syevj parameters.

4.2.1 Handle set-up and tear-down

List of handle initialization functions

- *hipsolverCreate()*
- *hipsolverDestroy()*

hipsolverCreate()

hipsolverStatus_t **hipsolverCreate**(*hipsolverHandle_t* *handle)

hipsolverDestroy()

hipsolverStatus_t **hipsolverDestroy**(*hipsolverHandle_t* handle)

4.2.2 Stream manipulation

List of stream manipulation functions

- *hipsolverSetStream()*
- *hipsolverGetStream()*

hipsolverSetStream()

hipsolverStatus_t **hipsolverSetStream**(*hipsolverHandle_t* handle, hipStream_t streamId)

hipsolverGetStream()

hipsolverStatus_t **hipsolverGetStream**(*hipsolverHandle_t* handle, hipStream_t *streamId)

4.2.3 Gesvdj parameter manipulation

List of gesvdj parameter functions

- *hipsolverCreateGesvdjInfo()*
- *hipsolverDestroyGesvdjInfo()*
- *hipsolverXgesvdjSetMaxSweeps()*
- *hipsolverXgesvdjSetSortEig()*
- *hipsolverXgesvdjSetTolerance()*
- *hipsolverXgesvdjGetResidual()*
- *hipsolverXgesvdjGetSweeps()*

hipsolverCreateGesvdjInfo()

hipsolverStatus_t **hipsolverCreateGesvdjInfo**(*hipsolverGesvdjInfo_t* *info)

hipsolverDestroyGesvdjInfo()

hipsolverStatus_t **hipsolverDestroyGesvdjInfo**(*hipsolverGesvdjInfo_t* info)

hipsolverXgesvdjSetMaxSweeps()

hipsolverStatus_t **hipsolverXgesvdjSetMaxSweeps**(*hipsolverGesvdjInfo_t* info, int max_sweeps)

hipsolverXgesvdjSetSortEig()

hipsolverStatus_t **hipsolverXgesvdjSetSortEig**(*hipsolverGesvdjInfo_t* info, int sort_eig)

hipsolverXgesvdjSetTolerance()

hipsolverStatus_t **hipsolverXgesvdjSetTolerance**(*hipsolverGesvdjInfo_t* info, double tolerance)

hipsolverXgesvdjGetResidual()

hipsolverStatus_t **hipsolverXgesvdjGetResidual**(*hipsolverHandle_t* handle, *hipsolverGesvdjInfo_t* info, double *residual)

hipsolverXgesvdjGetSweeps()

hipsolverStatus_t **hipsolverXgesvdjGetSweeps**(*hipsolverHandle_t* handle, *hipsolverGesvdjInfo_t* info, int *executed_sweeps)

4.2.4 Syevj parameter manipulation

List of syevj parameter functions

- *hipsolverCreateSyevjInfo()*
- *hipsolverDestroySyevjInfo()*
- *hipsolverXsyevjSetMaxSweeps()*
- *hipsolverXsyevjSetSortEig()*
- *hipsolverXsyevjSetTolerance()*
- *hipsolverXsyevjGetResidual()*
- *hipsolverXsyevjGetSweeps()*

hipsolverCreateSyevjInfo()

hipsolverStatus_t **hipsolverCreateSyevjInfo**(*hipsolverSyevjInfo_t* *info)

hipsolverDestroySyevjInfo()

hipsolverStatus_t **hipsolverDestroySyevjInfo**(*hipsolverSyevjInfo_t* info)

hipsolverXsyevjSetMaxSweeps()

hipsolverStatus_t **hipsolverXsyevjSetMaxSweeps**(*hipsolverSyevjInfo_t* info, int max_sweeps)

hipsolverXsyevjSetSortEig()

hipsolverStatus_t **hipsolverXsyevjSetSortEig**(*hipsolverSyevjInfo_t* info, int sort_eig)

hipsolverXsyevjSetTolerance()

hipsolverStatus_t **hipsolverXsyevjSetTolerance**(*hipsolverSyevjInfo_t* info, double tolerance)

hipsolverXsyevjGetResidual()

hipsolverStatus_t **hipsolverXsyevjGetResidual**(*hipsolverHandle_t* handle, *hipsolverSyevjInfo_t* info, double *residual)

hipsolverXsyevjGetSweeps()

hipsolverStatus_t **hipsolverXsyevjGetSweeps**(*hipsolverHandle_t* handle, *hipsolverSyevjInfo_t* info, int *executed_sweeps)

4.3 hipSOLVER LAPACK auxiliary functions

These are functions that support more *advanced LAPACK routines*. The auxiliary functions are divided into the following categories:

- *Orthonormal matrices*. Generation and application of orthonormal matrices.
- *Unitary matrices*. Generation and application of unitary matrices.

4.3.1 Orthonormal matrices

List of functions for orthonormal matrices

- *hipsolver<type>orgbr_bufferSize()*
- *hipsolver<type>orgbr()*
- *hipsolver<type>orgqr_bufferSize()*
- *hipsolver<type>orgqr()*
- *hipsolver<type>orgtr_bufferSize()*
- *hipsolver<type>orgtr()*
- *hipsolver<type>ormqr_bufferSize()*
- *hipsolver<type>ormqr()*
- *hipsolver<type>ormtr_bufferSize()*

- *hipsolver<type>ormtr()*

hipsolver<type>orgbr_bufferSize()

hipsolverStatus_t **hipsolverDorgbr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, double *A, int lda, double *tau, int *lwork)

hipsolverStatus_t **hipsolverSorgbr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, float *A, int lda, float *tau, int *lwork)

hipsolver<type>orgbr()

hipsolverStatus_t **hipsolverDorgbr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, double *A, int lda, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSorgbr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, float *A, int lda, float *tau, float *work, int lwork, int *devInfo)

hipsolver<type>orgqr_bufferSize()

hipsolverStatus_t **hipsolverDorgqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, double *A, int lda, double *tau, int *lwork)

hipsolverStatus_t **hipsolverSorgqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, float *A, int lda, float *tau, int *lwork)

hipsolver<type>orgqr()

hipsolverStatus_t **hipsolverDorgqr**(*hipsolverHandle_t* handle, int m, int n, int k, double *A, int lda, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSorgqr**(*hipsolverHandle_t* handle, int m, int n, int k, float *A, int lda, float *tau, float *work, int lwork, int *devInfo)

hipsolver<type>orgtr_bufferSize()

hipsolverStatus_t **hipsolverDorgtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *tau, int *lwork)

hipsolverStatus_t **hipsolverSorgtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *tau, int *lwork)

hipsolver<type>orgtr()

hipsolverStatus_t **hipsolverDorgtr**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSorgtr**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *tau, float *work, int lwork, int *devInfo)

hipsolver<type>ormqr_bufferSize()

hipsolverStatus_t **hipsolverDormqr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, double *A, int lda, double *tau, double *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverSormqr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, float *A, int lda, float *tau, float *C, int ldc, int *lwork)

hipsolver<type>ormqr()

hipsolverStatus_t **hipsolverDormqr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, double *A, int lda, double *tau, double *C, int ldc, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSormqr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, float *A, int lda, float *tau, float *C, int ldc, float *work, int lwork, int *devInfo)

hipsolver<type>ormtr_bufferSize()

hipsolverStatus_t **hipsolverDormtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, double *A, int lda, double *tau, double *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverSormtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, float *A, int lda, float *tau, float *C, int ldc, int *lwork)

hipsolver<type>ormtr()

hipsolverStatus_t **hipsolverDormtr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, double *A, int lda, double *tau, double *C, int ldc, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSormtr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, float *A, int lda, float *tau, float *C, int ldc, float *work, int lwork, int *devInfo)

4.3.2 Unitary matrices

List of functions for unitary matrices

- *hipsolver<type>ungbr_bufferSize()*
- *hipsolver<type>ungbr()*
- *hipsolver<type>ungqr_bufferSize()*
- *hipsolver<type>ungqr()*
- *hipsolver<type>ungtr_bufferSize()*

- *hipsolver<type>ungtr()*
- *hipsolver<type>unmqr_bufferSize()*
- *hipsolver<type>unmqr()*
- *hipsolver<type>unmtr_bufferSize()*
- *hipsolver<type>unmtr()*

hipsolver<type>ungbr_bufferSize()

hipsolverStatus_t **hipsolverZungbr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverCungbr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, int *lwork)

hipsolver<type>ungbr()

hipsolverStatus_t **hipsolverZungbr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCungbr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>ungqr_bufferSize()

hipsolverStatus_t **hipsolverZungqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverCungqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, int *lwork)

hipsolver<type>ungqr()

hipsolverStatus_t **hipsolverZungqr**(*hipsolverHandle_t* handle, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCungqr**(*hipsolverHandle_t* handle, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>ungtr_bufferSize()

hipsolverStatus_t **hipsolverZungtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverCungtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, int *lwork)

hipsolver<type>ungtr()

hipsolverStatus_t **hipsolverZungtr**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCungtr**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>unmqr_bufferSize()

hipsolverStatus_t **hipsolverZunmqr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverCunmqr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *C, int ldc, int *lwork)

hipsolver<type>unmqr()

hipsolverStatus_t **hipsolverZunmqr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCunmqr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverOperation_t* trans, int m, int n, int k, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *C, int ldc, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>unmtr_bufferSize()

hipsolverStatus_t **hipsolverZunmtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverCunmtr_bufferSize**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *C, int ldc, int *lwork)

hipsolver<type>unmtr()

hipsolverStatus_t **hipsolverZunmtr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCunmtr**(*hipsolverHandle_t* handle, *hipsolverSideMode_t* side, *hipsolverFillMode_t* uplo, *hipsolverOperation_t* trans, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *C, int ldc, hipFloatComplex *work, int lwork, int *devInfo)

4.4 hipSOLVER LAPACK Functions

LAPACK routines solve complex Numerical Linear Algebra problems. These functions are organized in the following categories:

- *Triangular factorizations*. Based on Gaussian elimination.
- *Orthogonal factorizations*. Based on Householder reflections.
- *Problem and matrix reductions*. Transformation of matrices and problems into equivalent forms.
- *Linear-systems solvers*. Based on triangular factorizations.
- *Least-squares solvers*. Based on orthogonal factorizations.
- *Symmetric eigensolvers*. Eigenproblems for symmetric matrices.
- *Singular value decomposition*. Singular values and related problems for general matrices.

4.4.1 Triangular factorizations

List of triangular factorizations

- *hipsolver<type>potrf_bufferSize()*
- *hipsolver<type>potrfBatched_bufferSize()*
- *hipsolver<type>potrf()*
- *hipsolver<type>potrfBatched()*
- *hipsolver<type>getrf_bufferSize()*
- *hipsolver<type>getrf()*
- *hipsolver<type>sytrf_bufferSize()*
- *hipsolver<type>sytrf()*

hipsolver<type>potrf_bufferSize()

hipsolverStatus_t **hipsolverZpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverCpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverSpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, int *lwork)

hipsolver<type>potrfBatched_bufferSize()

hipsolverStatus_t **hipsolverZpotrfBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A[], int lda, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverCpotrfBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A[], int lda, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverDpotrfBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A[], int lda, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverSpotrfBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A[], int lda, int *lwork, int batch_count)

hipsolver<type>potrf()

hipsolverStatus_t **hipsolverZpotrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCpotrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDpotrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSpotrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *work, int lwork, int *devInfo)

hipsolver<type>potrfBatched()

hipsolverStatus_t **hipsolverZpotrfBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A[], int lda, hipDoubleComplex *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverCpotrfBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A[], int lda, hipFloatComplex *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDpotrfBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A[], int lda, double *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverSpotrfBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A[], int lda, float *work, int lwork, int *devInfo, int batch_count)

hipsolver<type>getrf_bufferSize()

hipsolverStatus_t **hipsolverZgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverCgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverSgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, int *lwork)

hipsolver<type>getrf()

hipsolverStatus_t **hipsolverZgetrf**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int lwork, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverCgetrf**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int lwork, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverDgetrf**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *work, int lwork, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverSgetrf**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *work, int lwork, int *devIpiv, int *devInfo)

hipsolver<type>sytrf_bufferSize()

hipsolverStatus_t **hipsolverZsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverCsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverSsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, float *A, int lda, int *lwork)

hipsolver<type>sytrf()

hipsolverStatus_t **hipsolverZsytrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *ipiv, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCsytrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *ipiv, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDsytrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, int *ipiv, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsytrf**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, int *ipiv, float *work, int lwork, int *devInfo)

4.4.2 Orthogonal factorizations

List of orthogonal factorizations

- *hipsolver<type>geqrf_bufferSize()*
- *hipsolver<type>geqrf()*

hipsolver<type>geqrf_bufferSize()

hipsolverStatus_t **hipsolverZgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverCgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverSgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, int *lwork)

hipsolver<type>geqrf()

hipsolverStatus_t **hipsolverZgeqrf**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCgeqrf**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDgeqrf**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSgeqrf**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *tau, float *work, int lwork, int *devInfo)

4.4.3 Problem and matrix reductions

List of reductions

- *hipsolver<type>gebrd_bufferSize()*
- *hipsolver<type>gebrd()*
- *hipsolver<type>sytrd_bufferSize()*
- *hipsolver<type>hetrd_bufferSize()*
- *hipsolver<type>sytrd()*
- *hipsolver<type>hetrd()*

hipsolver<type>gebrd_bufferSize()

hipsolverStatus_t **hipsolverZgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverCgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverSgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolver<type>gebrd()

hipsolverStatus_t **hipsolverZgebrd**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, double *D, double *E, hipDoubleComplex *tauq, hipDoubleComplex *taup, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCgebrd**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, float *D, float *E, hipFloatComplex *tauq, hipFloatComplex *taup, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDgebrd**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *D, double *E, double *tauq, double *taup, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSgebrd**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *D, float *E, float *tauq, float *taup, float *work, int lwork, int *devInfo)

hipsolver<type>sytrd_bufferSize()

hipsolverStatus_t **hipsolverDsytrd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *D, double *E, double *tau, int *lwork)

hipsolverStatus_t **hipsolverSsytrd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *D, float *E, float *tau, int *lwork)

hipsolver<type>hetrd_bufferSize()

hipsolverStatus_t **hipsolverZhetrd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *D, double *E, hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverChetrd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *D, float *E, hipFloatComplex *tau, int *lwork)

hipsolver<type>sytrd()

hipsolverStatus_t **hipsolverDsytrd**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *D, double *E, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsytrd**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *D, float *E, float *tau, float *work, int lwork, int *devInfo)

hipsolver<type>hetrd()

hipsolverStatus_t **hipsolverZhetrd**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *D, double *E, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverChetrd**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *D, float *E, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

4.4.4 Linear-systems solvers

List of linear solvers

- *hipsolver<type>potri_bufferSize()*
- *hipsolver<type>potri()*
- *hipsolver<type>potrs_bufferSize()*
- *hipsolver<type>potrsBatched_bufferSize()*
- *hipsolver<type>potrs()*
- *hipsolver<type>potrsBatched()*
- *hipsolver<type>getrs_bufferSize()*
- *hipsolver<type>getrs()*
- *hipsolver<type><type>gesv_bufferSize()*
- *hipsolver<type><type>gesv()*

hipsolver<type>potri_bufferSize()

hipsolverStatus_t **hipsolverZpotri_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverCpotri_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDpotri_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverSpotri_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, int *lwork)

hipsolver<type>potri()

hipsolverStatus_t **hipsolverZpotri**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCpotri**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDpotri**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSpotri**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *work, int lwork, int *devInfo)

hipsolver<type>potrs_bufferSize()

hipsolverStatus_t **hipsolverZpotrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverCpotrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverDpotrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, double *A, int lda, double *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverSpotrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, float *A, int lda, float *B, int ldb, int *lwork)

hipsolver<type>potrsBatched_bufferSize()

hipsolverStatus_t **hipsolverZpotrsBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipDoubleComplex *A[], int lda, hipDoubleComplex *B[], int ldb, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverCpotrsBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipFloatComplex *A[], int lda, hipFloatComplex *B[], int ldb, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverDpotrsBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, double *A[], int lda, double *B[], int ldb, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverSpotrsBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, float *A[], int lda, float *B[], int ldb, int *lwork, int batch_count)

hipsolver<type>potrs()

hipsolverStatus_t **hipsolverZpotrs**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCpotrs**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDpotrs**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, double *A, int lda, double *B, int ldb, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSpotrs**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, float *A, int lda, float *B, int ldb, float *work, int lwork, int *devInfo)

hipsolver<type>potrsBatched()

hipsolverStatus_t **hipsolverZpotrsBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipDoubleComplex *A[], int lda, hipDoubleComplex *B[], int ldb, hipDoubleComplex *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverCpotrsBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, hipFloatComplex *A[], int lda, hipFloatComplex *B[], int ldb, hipFloatComplex *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDpotrsBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, double *A[], int lda, double *B[], int ldb, double *work, int lwork, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverSpotrsBatched**(*hipsolverHandle_t* handle, *hipsolverFillMode_t* uplo, int n, int nrhs, float *A[], int lda, float *B[], int ldb, float *work, int lwork, int *devInfo, int batch_count)

hipsolver<type>getrs_bufferSize()

hipsolverStatus_t **hipsolverZgetrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverCgetrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverDgetrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, int *lwork)

hipsolverStatus_t **hipsolverSgetrs_bufferSize**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, int *lwork)

hipsolver<type>getrs()

hipsolverStatus_t **hipsolverZgetrs**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCgetrs**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDgetrs**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSgetrs**(*hipsolverHandle_t* handle, *hipsolverOperation_t* trans, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, float *work, int lwork, int *devInfo)

hipsolver<type><type>gesv_bufferSize()

hipsolverStatus_t **hipsolverZZgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverCCgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverDDgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, double *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverSSgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, float *X, int ldx, size_t *lwork)

hipsolver<type><type>gesv()

hipsolverStatus_t **hipsolverZZgesv**(*hipsolverHandle_t* handle, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverCCgesv**(*hipsolverHandle_t* handle, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDDgesv**(*hipsolverHandle_t* handle, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, double *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverSSgesv**(*hipsolverHandle_t* handle, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, float *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

4.4.5 Least-squares solvers

List of least-squares solvers

- *hipsolver<type><type>gels_bufferSize()*
- *hipsolver<type><type>gels()*

hipsolver<type><type>gels_bufferSize()

hipsolverStatus_t **hipsolverZZgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverCCgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverDDgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, double *A, int lda, double *B, int ldb, double *X, int ldx, size_t *lwork)

hipsolverStatus_t **hipsolverSSgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, float *A, int lda, float *B, int ldb, float *X, int ldx, size_t *lwork)

hipsolver<type><type>gels()

hipsolverStatus_t **hipsolverZZgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverCCgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDDgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, double *A, int lda, double *B, int ldb, double *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverSSgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, float *A, int lda, float *B, int ldb, float *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

4.4.6 Symmetric eigensolvers

List of symmetric eigensolvers

- *hipsolver<type>syevd_bufferSize()*
- *hipsolver<type>heevd_bufferSize()*
- *hipsolver<type>syevd()*
- *hipsolver<type>heevd()*
- *hipsolver<type>sygvd_bufferSize()*
- *hipsolver<type>hegvd_bufferSize()*
- *hipsolver<type>sygvd()*
- *hipsolver<type>hegvd()*

hipsolver<type>syevd_bufferSize()

hipsolverStatus_t **hipsolverDsyevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *D, int *lwork)

hipsolverStatus_t **hipsolverSsyevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *D, int *lwork)

hipsolver<type>heevd_bufferSize()

hipsolverStatus_t **hipsolverZheevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *D, int *lwork)

hipsolverStatus_t **hipsolverCheevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *D, int *lwork)

hipsolver<type>syevd()

hipsolverStatus_t **hipsolverDsyevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *D, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsyevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *D, float *work, int lwork, int *devInfo)

hipsolver<type>heevd()

hipsolverStatus_t **hipsolverZheevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *D, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCheevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *D, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>sygvd_bufferSize()

hipsolverStatus_t **hipsolverDsygvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, int *lwork)

hipsolverStatus_t **hipsolverSsygvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, int *lwork)

hipsolver<type>hegvd_bufferSize()

hipsolverStatus_t **hipsolverZhegvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, int *lwork)

hipsolverStatus_t **hipsolverChegvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, int *lwork)

hipsolver<type>sygvd()

hipsolverStatus_t **hipsolverDsygvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsygvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, float *work, int lwork, int *devInfo)

hipsolver<type>hegvd()

hipsolverStatus_t **hipsolverZhegvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverChegvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, hipFloatComplex *work, int lwork, int *devInfo)

4.4.7 Singular value decomposition

List of SVD related functions

- *hipsolver<type>gesvd_bufferSize()*
- *hipsolver<type>gesvd()*

hipsolver<type>gesvd_bufferSize()

hipsolverStatus_t **hipsolverZgesvd_bufferSize**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverCgesvd_bufferSize**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDgesvd_bufferSize**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverSgesvd_bufferSize**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, int *lwork)

hipsolver<type>gesvd()

hipsolverStatus_t **hipsolverZgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, double *rwork, int *devInfo)

hipsolverStatus_t **hipsolverCgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, float *rwork, int *devInfo)

hipsolverStatus_t **hipsolverDgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, double *rwork, int *devInfo)

hipsolverStatus_t **hipsolverSgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, float *rwork, int *devInfo)

4.5 hipSOLVER LAPACK-like functions

Other Lapack-like routines provided by hipSOLVER are divided into the following subcategories:

- *Symmetric eigensolvers*. Eigenproblems for symmetric matrices.
- *Singular value decomposition*. Singular values and related problems for general matrices.

4.5.1 Symmetric eigensolvers

List of Lapack-like symmetric eigensolvers

- *hipsolver<type>syevdx_bufferSize()*
- *hipsolver<type>heevdx_bufferSize()*
- *hipsolver<type>syevdx()*
- *hipsolver<type>heevdx()*
- *hipsolver<type>syevj_bufferSize()*
- *hipsolver<type>heevj_bufferSize()*
- *hipsolver<type>syevjBatched_bufferSize()*
- *hipsolver<type>heevjBatched_bufferSize()*
- *hipsolver<type>syevj()*
- *hipsolver<type>heevj()*
- *hipsolver<type>syevjBatched()*
- *hipsolver<type>heevjBatched()*
- *hipsolver<type>sygvdx_bufferSize()*
- *hipsolver<type>hegvdx_bufferSize()*
- *hipsolver<type>sygvdx()*
- *hipsolver<type>hegvdx()*
- *hipsolver<type>sygvj_bufferSize()*
- *hipsolver<type>hegvj_bufferSize()*

- *hipsolver<type>sygvj()*
- *hipsolver<type>hegvj()*

hipsolver<type>syevdx_bufferSize()

hipsolverStatus_t **hipsolverDsyevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const double *A, int lda, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverSsyevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const float *A, int lda, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolver<type>heevdx_bufferSize()

hipsolverStatus_t **hipsolverZheevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverCheevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolver<type>syevdx()

hipsolverStatus_t **hipsolverDsyevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, double *A, int lda, double vl, double vu, int il, int iu, int *nev, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsyevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, float *A, int lda, float vl, float vu, int il, int iu, int *nev, float *W, float *work, int lwork, int *devInfo)

hipsolver<type>heevdx()

hipsolverStatus_t **hipsolverZheevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double vl, double vu, int il, int iu, int *nev, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverCheevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float vl, float vu, int il, int iu, int *nev, float *W, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>syevj_bufferSize()

hipsolverStatus_t **hipsolverDsyevj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverSsyevj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolver<type>heevj_bufferSize()

hipsolverStatus_t **hipsolverZheevj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverCheevj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolver<type>syevjBatched_bufferSize()

hipsolverStatus_t **hipsolverDsyevjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverSsyevjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolver<type>heevjBatched_bufferSize()

hipsolverStatus_t **hipsolverZheevjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverCheevjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolver<type>syevj()

hipsolverStatus_t **hipsolverDsyevj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverSsyevj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolver<type>heevj()

hipsolverStatus_t **hipsolverZheevj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverCheevj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolver<type>syevjBatched()

hipsolverStatus_t **hipsolverDsyevjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverSsyevjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolver<type>heevjBatched()

hipsolverStatus_t **hipsolverZheevjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverCheevjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolver<type>sygvdx_bufferSize()

hipsolverStatus_t **hipsolverDsygvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *B, int ldb, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverSsygvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *B, int ldb, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolver<type>hegvdx_bufferSize()

hipsolverStatus_t **hipsolverZhegvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *B, int ldb, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverChegvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *B, int ldb, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolver<type>sygvdx()

hipsolverStatus_t **hipsolverDsygvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double vl, double vu, int il, int iu, int *nev, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverSsygvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float vl, float vu, int il, int iu, int *nev, float *W, float *work, int lwork, int *devInfo)

hipsolver<type>hegvdx()

hipsolverStatus_t **hipsolverZhegvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double vl, double vu, int il, int iu, int *nev, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverChegvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float vl, float vu, int il, int iu, int *nev, float *W, hipFloatComplex *work, int lwork, int *devInfo)

hipsolver<type>sygvj_bufferSize()

hipsolverStatus_t **hipsolverDsygvj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverSsygvj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolver<type>hegvj_bufferSize()

hipsolverStatus_t **hipsolverZhegvj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverChegvj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolver<type>sygvj()

hipsolverStatus_t **hipsolverDsygvj**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverSsygvj**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolver<type>hegvj()

hipsolverStatus_t **hipsolverZhegvj**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverChegvj**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

4.5.2 Singular value decomposition

List of Lapack-like SVD related functions

- *hipsolver<type>gesvdj_bufferSize()*
- *hipsolver<type>gesvdjBatched_bufferSize()*
- *hipsolver<type>gesvdj()*
- *hipsolver<type>gesvdjBatched()*

hipsolver<type>gesvdj_bufferSize()

hipsolverStatus_t **hipsolverZgesvdj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const hipDoubleComplex *A, int lda, const double *S, const hipDoubleComplex *U, int ldu, const hipDoubleComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverCgesvdj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const hipFloatComplex *A, int lda, const float *S, const hipFloatComplex *U, int ldu, const hipFloatComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDgesvdj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const double *A, int lda, const double *S, const double *U, int ldu, const double *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverSgesvdj_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const float *A, int lda, const float *S, const float *U, int ldu, const float *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolver<type>gesvdjBatched_bufferSize()

hipsolverStatus_t **hipsolverZgesvdjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const hipDoubleComplex *A, int lda, const double *S, const hipDoubleComplex *U, int ldu, const hipDoubleComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverCgesvdjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const hipFloatComplex *A, int lda, const float *S, const hipFloatComplex *U, int ldu, const hipFloatComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDgesvdjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const double *A, int lda, const double *S, const double *U, int ldu, const double *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverSgesvdjBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const float *A, int lda, const float *S, const float *U, int ldu, const float *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolver<type>gesvdj()

hipsolverStatus_t **hipsolverZgesvdj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverCgesvdj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDgesvdj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverSgesvdj**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolver<type>gesvdjBatched()

hipsolverStatus_t **hipsolverZgesvdjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverCgesvdjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDgesvdjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverSgesvdjBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

CHAPTER

FIVE

HIPSOLVER COMPATIBILITY API - DENSE MATRICES

This document provides the method signatures for the wrapper functions that are currently implemented in hipSOLVER. For a complete description of the functions' behavior and arguments, see the corresponding backend documentation at cuSOLVER API and/or rocSOLVER API.

For ease of porting from existing cuSOLVER applications to hipSOLVER, functions in the hipsolverDn compatibility API are designed to have method signatures that are consistent with the cusolverDn interface. However, *performance issues* may arise when using the rocSOLVER backend due to differing workspace requirements. Therefore, users interested in achieving the best performance with the rocSOLVER backend should consult the *regular API documentation*, and transition from the compatibility API to the regular API at the earliest convenience. Please refer to *Using hipSOLVER* for additional *considerations regarding the use of the compatibility API*.

5.1 Dense matrix datatypes

hipSOLVER defines types and enumerations that are internally converted to the corresponding backend types at runtime. Here we list the types used in this compatibility API.

5.1.1 hipsolverDnHandle_t

typedef *hipsolverHandle_t* **hipsolverDnHandle_t**

Provided for convenience when porting code from cuSOLVER.

5.1.2 hipsolverGesvdjInfo_t

See *hipsolverGesvdjInfo_t*.

5.1.3 hipsolverSyevjInfo_t

See *hipsolverSyevjInfo_t*.

5.1.4 hipsolverStatus_t

See *hipsolverStatus_t*.

5.1.5 hipblasOperation_t

See *hipblasOperation_t*.

5.1.6 hipblasFillMode_t

See *hipblasFillMode_t*.

5.1.7 hipblasSideMode_t

See *hipblasSideMode_t*.

5.1.8 hipsolverEigMode_t

See *hipsolverEigMode_t*.

5.1.9 hipsolverEigType_t

See *hipsolverEigType_t*.

5.1.10 hipsolverEigRange_t

See *hipsolverEigRange_t*.

5.1.11 hipsolverAlgMode_t

enum **hipsolverAlgMode_t**

Values:

enumerator **HIPSOLVER_ALG_0**

enumerator **HIPSOLVER_ALG_1**

5.1.12 hipsolverDnFunction_t

enum **hipsolverDnFunction_t**

Values:

enumerator **HIPSOLVERDN_GETRF**

5.2 Dense matrix helper functions

These are helper functions that control aspects of the hipSOLVER library. They are divided into the following categories:

- *Handle set-up and tear-down* functions used to initialize and cleanup the library handle
- *Stream manipulation* functions provide functionality to manipulate streams

- *Gesvdj parameter manipulation* functions provide functionality to manipulate gesvdj parameters
- *Syevj parameter manipulation* functions provide functionality to manipulate syevj parameters
- *Other parameter manipulation* functions provide functionality to manipulate other parameters

5.2.1 Handle set-up and tear-down

List of handle initialization functions

- *hipsolverDnCreate()*
- *hipsolverDnDestroy()*

hipsolverDnCreate()

hipsolverStatus_t **hipsolverDnCreate**(*hipsolverHandle_t* *handle)

An alias for *hipsolverCreate*.

hipsolverDnDestroy()

hipsolverStatus_t **hipsolverDnDestroy**(*hipsolverHandle_t* handle)

An alias for *hipsolverDestroy*.

5.2.2 Stream manipulation

List of stream manipulation functions

- *hipsolverDnSetStream()*
- *hipsolverDnGetStream()*

hipsolverDnSetStream()

hipsolverStatus_t **hipsolverDnSetStream**(*hipsolverHandle_t* handle, hipStream_t streamId)

An alias for *hipsolverSetStream*.

hipsolverDnGetStream()

hipsolverStatus_t **hipsolverDnGetStream**(*hipsolverHandle_t* handle, hipStream_t *streamId)

An alias for *hipsolverGetStream*.

5.2.3 Gesvdj parameter manipulation

List of gesvdj parameter functions

- *hipsolverDnCreateGesvdjInfo()*
- *hipsolverDnDestroyGesvdjInfo()*

- *hipsolverDnXgesvdjSetMaxSweeps()*
- *hipsolverDnXgesvdjSetSortEig()*
- *hipsolverDnXgesvdjSetTolerance()*
- *hipsolverDnXgesvdjGetResidual()*
- *hipsolverDnXgesvdjGetSweeps()*

hipsolverDnCreateGesvdjInfo()

hipsolverStatus_t **hipsolverDnCreateGesvdjInfo**(*hipsolverGesvdjInfo_t* *info)

hipsolverDnDestroyGesvdjInfo()

hipsolverStatus_t **hipsolverDnDestroyGesvdjInfo**(*hipsolverGesvdjInfo_t* info)

hipsolverDnXgesvdjSetMaxSweeps()

hipsolverStatus_t **hipsolverDnXgesvdjSetMaxSweeps**(*hipsolverGesvdjInfo_t* info, int max_sweeps)

hipsolverDnXgesvdjSetSortEig()

hipsolverStatus_t **hipsolverDnXgesvdjSetSortEig**(*hipsolverGesvdjInfo_t* info, int sort_eig)

hipsolverDnXgesvdjSetTolerance()

hipsolverStatus_t **hipsolverDnXgesvdjSetTolerance**(*hipsolverGesvdjInfo_t* info, double tolerance)

hipsolverDnXgesvdjGetResidual()

hipsolverStatus_t **hipsolverDnXgesvdjGetResidual**(*hipsolverDnHandle_t* handle, *hipsolverGesvdjInfo_t* info, double *residual)

hipsolverDnXgesvdjGetSweeps()

hipsolverStatus_t **hipsolverDnXgesvdjGetSweeps**(*hipsolverDnHandle_t* handle, *hipsolverGesvdjInfo_t* info, int *executed_sweeps)

5.2.4 Syevj parameter manipulation

List of syevj parameter functions

- *hipsolverDnCreateSyevjInfo()*
- *hipsolverDnDestroySyevjInfo()*
- *hipsolverDnXsyevjSetMaxSweeps()*
- *hipsolverDnXsyevjSetSortEig()*
- *hipsolverDnXsyevjSetTolerance()*

- *hipsolverDnXsyevjGetResidual()*
- *hipsolverDnXsyevjGetSweeps()*

hipsolverDnCreateSyevjInfo()

hipsolverStatus_t **hipsolverDnCreateSyevjInfo**(*hipsolverSyevjInfo_t* *info)

hipsolverDnDestroySyevjInfo()

hipsolverStatus_t **hipsolverDnDestroySyevjInfo**(*hipsolverSyevjInfo_t* info)

hipsolverDnXsyevjSetMaxSweeps()

hipsolverStatus_t **hipsolverDnXsyevjSetMaxSweeps**(*hipsolverSyevjInfo_t* info, int max_sweeps)

hipsolverDnXsyevjSetSortEig()

hipsolverStatus_t **hipsolverDnXsyevjSetSortEig**(*hipsolverSyevjInfo_t* info, int sort_eig)

hipsolverDnXsyevjSetTolerance()

hipsolverStatus_t **hipsolverDnXsyevjSetTolerance**(*hipsolverSyevjInfo_t* info, double tolerance)

hipsolverDnXsyevjGetResidual()

hipsolverStatus_t **hipsolverDnXsyevjGetResidual**(*hipsolverDnHandle_t* handle, *hipsolverSyevjInfo_t* info, double *residual)

hipsolverDnXsyevjGetSweeps()

hipsolverStatus_t **hipsolverDnXsyevjGetSweeps**(*hipsolverDnHandle_t* handle, *hipsolverSyevjInfo_t* info, int *executed_sweeps)

5.2.5 Other parameter manipulation

List of other parameter functions

- *hipsolverDnCreateParams()*
- *hipsolverDnDestroyParams()*
- *hipsolverDnSetAdvOptions()*

hipsolverDnCreateParams()

hipsolverStatus_t **hipsolverDnCreateParams**(hipsolverDnParams_t *params)

hipsolverDnDestroyParams()

hipsolverStatus_t **hipsolverDnDestroyParams**(hipsolverDnParams_t params)

hipsolverDnSetAdvOptions()

hipsolverStatus_t **hipsolverDnSetAdvOptions**(hipsolverDnParams_t params, *hipsolverDnFunction_t* func, *hipsolverAlgMode_t* alg)

5.3 Dense matrix LAPACK auxiliary functions

These are functions that support more *advanced LAPACK routines*. The auxiliary functions are divided into the following categories:

- *Orthonormal matrices*. Generation and application of orthonormal matrices.
- *Unitary matrices*. Generation and application of unitary matrices.

5.3.1 Orthonormal matrices

List of functions for orthonormal matrices

- *hipsolverDn<type>orgbr_bufferSize()*
- *hipsolverDn<type>orgbr()*
- *hipsolverDn<type>orgqr_bufferSize()*
- *hipsolverDn<type>orgqr()*
- *hipsolverDn<type>orgtr_bufferSize()*
- *hipsolverDn<type>orgtr()*
- *hipsolverDn<type>ormqr_bufferSize()*
- *hipsolverDn<type>ormqr()*
- *hipsolverDn<type>ormtr_bufferSize()*
- *hipsolverDn<type>ormtr()*

hipsolverDn<type>orgbr_bufferSize()

hipsolverStatus_t **hipsolverDnDorgbr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, const double *A, int lda, const double *tau, int *lwork)

hipsolverStatus_t **hipsolverDnSorgbr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, const float *A, int lda, const float *tau, int *lwork)

hipsolverDn<type>orgbr()

hipsolverStatus_t **hipsolverDnDorgbr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, double *A, int lda, const double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSorgbr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, float *A, int lda, const float *tau, float *work, int lwork, int *devInfo)

hipsolverDn<type>orgqr_bufferSize()

hipsolverStatus_t **hipsolverDnDorgqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, const double *A, int lda, const double *tau, int *lwork)

hipsolverStatus_t **hipsolverDnSorgqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, const float *A, int lda, const float *tau, int *lwork)

hipsolverDn<type>orgqr()

hipsolverStatus_t **hipsolverDnDorgqr**(*hipsolverHandle_t* handle, int m, int n, int k, double *A, int lda, const double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSorgqr**(*hipsolverHandle_t* handle, int m, int n, int k, float *A, int lda, const float *tau, float *work, int lwork, int *devInfo)

hipsolverDn<type>orgtr_bufferSize()

hipsolverStatus_t **hipsolverDnDorgtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *tau, int *lwork)

hipsolverStatus_t **hipsolverDnSorgtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *tau, int *lwork)

hipsolverDn<type>orgtr()

hipsolverStatus_t **hipsolverDnDorgtr**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, const double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSorgtr**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, const float *tau, float *work, int lwork, int *devInfo)

hipsolverDn<type>ormqr_bufferSize()

hipsolverStatus_t **hipsolverDnDormqr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const double *A, int lda, const double *tau, const double *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverDnSormqr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const float *A, int lda, const float *tau, const float *C, int ldc, int *lwork)

hipsolverDn<type>ormqr()

hipsolverStatus_t **hipsolverDnDormqr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const double *A, int lda, const double *tau, double *C, int ldc, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSormqr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const float *A, int lda, const float *tau, float *C, int ldc, float *work, int lwork, int *devInfo)

hipsolverDn<type>ormtr_bufferSize()

hipsolverStatus_t **hipsolverDnDormtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, const double *A, int lda, const double *tau, const double *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverDnSormtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, const float *A, int lda, const float *tau, const float *C, int ldc, int *lwork)

hipsolverDn<type>ormtr()

hipsolverStatus_t **hipsolverDnDormtr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, double *A, int lda, double *tau, double *C, int ldc, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSormtr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, float *A, int lda, float *tau, float *C, int ldc, float *work, int lwork, int *devInfo)

5.3.2 Unitary matrices

List of functions for unitary matrices

- *hipsolverDn<type>ungbr_bufferSize()*
- *hipsolverDn<type>ungbr()*
- *hipsolverDn<type>ungqr_bufferSize()*
- *hipsolverDn<type>ungqr()*
- *hipsolverDn<type>ungtr_bufferSize()*
- *hipsolverDn<type>ungtr()*
- *hipsolverDn<type>unmqr_bufferSize()*
- *hipsolverDn<type>unmqr()*
- *hipsolverDn<type>unmtr_bufferSize()*
- *hipsolverDn<type>unmtr()*

hipsolverDn<type>ungbr_bufferSize()

hipsolverStatus_t **hipsolverDnZungbr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverDnCungbr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, int *lwork)

hipsolverDn<type>ungbr()

hipsolverStatus_t **hipsolverDnZungbr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCungbr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, int m, int n, int k, hipFloatComplex *A, int lda, const hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>ungqr_bufferSize()

hipsolverStatus_t **hipsolverDnZungqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverDnCungqr_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int k, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, int *lwork)

hipsolverDn<type>ungqr()

hipsolverStatus_t **hipsolverDnZungqr**(*hipsolverHandle_t* handle, int m, int n, int k, hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCungqr**(*hipsolverHandle_t* handle, int m, int n, int k, hipFloatComplex *A, int lda, const hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>ungtr_bufferSize()

hipsolverStatus_t **hipsolverDnZungtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverDnCungtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, int *lwork)

hipsolverDn<type>ungtr()

hipsolverStatus_t **hipsolverDnZungtr**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCungtr**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, const hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>unmqr_bufferSize()

hipsolverStatus_t **hipsolverDnZunmqr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, const hipDoubleComplex *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverDnCunmqr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, const hipFloatComplex *C, int ldc, int *lwork)

hipsolverDn<type>unmqr()

hipsolverStatus_t **hipsolverDnZunmqr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCunmqr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasOperation_t* trans, int m, int n, int k, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, hipFloatComplex *C, int ldc, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>unmtr_bufferSize()

hipsolverStatus_t **hipsolverDnZunmtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *tau, const hipDoubleComplex *C, int ldc, int *lwork)

hipsolverStatus_t **hipsolverDnCunmtr_bufferSize**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *tau, const hipFloatComplex *C, int ldc, int *lwork)

hipsolverDn<type>unmtr()

hipsolverStatus_t **hipsolverDnZunmtr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *C, int ldc, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCunmtr**(*hipsolverHandle_t* handle, *hipblasSideMode_t* side, *hipblasFillMode_t* uplo, *hipblasOperation_t* trans, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *C, int ldc, hipFloatComplex *work, int lwork, int *devInfo)

5.4 Dense matrix LAPACK functions

LAPACK routines solve complex Numerical Linear Algebra problems. These functions are organized in the following categories:

- *Triangular factorizations*. Based on Gaussian elimination.
- *Orthogonal factorizations*. Based on Householder reflections.
- *Problem and matrix reductions*. Transformation of matrices and problems into equivalent forms.
- *Linear-systems solvers*. Based on triangular factorizations.

- *Least-squares solvers*. Based on orthogonal factorizations.
- *Symmetric eigensolvers*. Eigenproblems for symmetric matrices.
- *Singular value decomposition*. Singular values and related problems for general matrices.

5.4.1 Triangular factorizations

List of triangular factorizations

- *hipsolverDn<type>potrf_bufferSize()*
- *hipsolverDn<type>potrf()*
- *hipsolverDn<type>potrfBatched()*
- *hipsolverDn<type>getrf_bufferSize()*
- *hipsolverDn<type>getrf()*
- *hipsolverDn<type>sytrf_bufferSize()*
- *hipsolverDn<type>sytrf()*

hipsolverDn<type>potrf_bufferSize()

hipsolverStatus_t **hipsolverDnZpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnCpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnDpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnSpotrf_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, int *lwork)

hipsolverDn<type>potrf()

hipsolverStatus_t **hipsolverDnZpotrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCpotrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDpotrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSpotrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *work, int lwork, int *devInfo)

hipsolverDn<type>potrfBatched()

hipsolverStatus_t **hipsolverDnZpotrfBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A[], int lda, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnCpotrfBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A[], int lda, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnDpotrfBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A[], int lda, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnSpotrfBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A[], int lda, int *devInfo, int batch_count)

hipsolverDn<type>getrf_bufferSize()

hipsolverStatus_t **hipsolverDnXgetrf_bufferSize**(*hipsolverDnHandle_t* handle, hipsolverDnParams_t params, int64_t m, int64_t n, hipDataType dataTypeA, const void *A, int64_t lda, hipDataType computeType, size_t *lworkOnDevice, size_t *lworkOnHost)

hipsolverStatus_t **hipsolverDnZgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnCgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnDgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnSgetrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, int *lwork)

hipsolverDn<type>getrf()

hipsolverStatus_t **hipsolverDnXgetrf**(*hipsolverDnHandle_t* handle, hipsolverDnParams_t params, int64_t m, int64_t n, hipDataType dataTypeA, void *A, int64_t lda, int64_t *devIpiv, hipDataType computeType, void *workOnDevice, size_t lworkOnDevice, void *workOnHost, size_t lworkOnHost, int *devInfo)

hipsolverStatus_t **hipsolverDnZgetrf**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverDnCgetrf**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverDnDgetrf**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *work, int *devIpiv, int *devInfo)

hipsolverStatus_t **hipsolverDnSgetrf**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *work, int *devIpiv, int *devInfo)

hipsolverDn<type>sytrf_bufferSize()

hipsolverStatus_t **hipsolverDnZsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnCsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnDsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnSsytrf_bufferSize**(*hipsolverHandle_t* handle, int n, float *A, int lda, int *lwork)

hipsolverDn<type>sytrf()

hipsolverStatus_t **hipsolverDnZsytrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *ipiv, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCsytrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *ipiv, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDsytrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, int *ipiv, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsytrf**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, int *ipiv, float *work, int lwork, int *devInfo)

5.4.2 Orthogonal factorizations

List of orthogonal factorizations

- *hipsolverDn<type>geqrf_bufferSize()*
- *hipsolverDn<type>geqrf()*

hipsolverDn<type>geqrf_bufferSize()

hipsolverStatus_t **hipsolverDnZgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnCgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnDgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnSgeqrf_bufferSize**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, int *lwork)

hipsolverDn<type>geqrf()

hipsolverStatus_t **hipsolverDnZgeqrf**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCgeqrf**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDgeqrf**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSgeqrf**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *tau, float *work, int lwork, int *devInfo)

5.4.3 Problem and matrix reductions

List of reductions

- *hipsolverDn<type>gebrd_bufferSize()*
- *hipsolverDn<type>gebrd()*
- *hipsolverDn<type>sytrd_bufferSize()*
- *hipsolverDn<type>hetrd_bufferSize()*
- *hipsolverDn<type>sytrd()*
- *hipsolverDn<type>hetrd()*

hipsolverDn<type>gebrd_bufferSize()

hipsolverStatus_t **hipsolverDnZgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnCgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnDgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnSgebrd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverDn<type>gebrd()

hipsolverStatus_t **hipsolverDnZgebrd**(*hipsolverHandle_t* handle, int m, int n, hipDoubleComplex *A, int lda, double *D, double *E, hipDoubleComplex *tauq, hipDoubleComplex *taup, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCgebrd**(*hipsolverHandle_t* handle, int m, int n, hipFloatComplex *A, int lda, float *D, float *E, hipFloatComplex *tauq, hipFloatComplex *taup, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDgebrd**(*hipsolverHandle_t* handle, int m, int n, double *A, int lda, double *D, double *E, double *tauq, double *taup, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSgebrd**(*hipsolverHandle_t* handle, int m, int n, float *A, int lda, float *D, float *E, float *tauq, float *taup, float *work, int lwork, int *devInfo)

hipsolverDn<type>sytrd_bufferSize()

hipsolverStatus_t **hipsolverDnDsytrd_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *D, const double *E, const double *tau, int *lwork)

hipsolverStatus_t **hipsolverDnSsytrd_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *D, const float *E, const float *tau, int *lwork)

hipsolverDn<type>hetrd_bufferSize()

hipsolverStatus_t **hipsolverDnZhetrd_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const double *D, const double *E, const hipDoubleComplex *tau, int *lwork)

hipsolverStatus_t **hipsolverDnChetrd_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const float *D, const float *E, const hipFloatComplex *tau, int *lwork)

hipsolverDn<type>sytrd()

hipsolverStatus_t **hipsolverDnDsytrd**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *D, double *E, double *tau, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsytrd**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *D, float *E, float *tau, float *work, int lwork, int *devInfo)

hipsolverDn<type>hetrd()

hipsolverStatus_t **hipsolverDnZhetrd**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *D, double *E, hipDoubleComplex *tau, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnChetrd**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *D, float *E, hipFloatComplex *tau, hipFloatComplex *work, int lwork, int *devInfo)

5.4.4 Linear-systems solvers

List of linear solvers

- *hipsolverDn<type>potri_bufferSize()*
- *hipsolverDn<type>potri()*
- *hipsolverDn<type>potrs()*
- *hipsolverDn<type>potrsBatched()*
- *hipsolverDn<type>getrs()*
- *hipsolverDn<type><type>gesv_bufferSize()*
- *hipsolverDn<type><type>gesv()*

hipsolverDn<type>potri_bufferSize()

hipsolverStatus_t **hipsolverDnZpotri_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnCpotri_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnDpotri_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, int *lwork)

hipsolverStatus_t **hipsolverDnSpotri_bufferSize**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, int *lwork)

hipsolverDn<type>potri()

hipsolverStatus_t **hipsolverDnZpotri**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCpotri**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDpotri**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSpotri**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *work, int lwork, int *devInfo)

hipsolverDn<type>potrs()

hipsolverStatus_t **hipsolverDnZpotrs**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, const hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnCpotrs**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, const hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnDpotrs**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, const double *A, int lda, double *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnSpotrs**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, const float *A, int lda, float *B, int ldb, int *devInfo)

hipsolverDn<type>potrsBatched()

hipsolverStatus_t **hipsolverDnZpotrsBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, hipDoubleComplex *A[], int lda, hipDoubleComplex *B[], int ldb, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnCpotrsBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, hipFloatComplex *A[], int lda, hipFloatComplex *B[], int ldb, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnDpotrsBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, double *A[], int lda, double *B[], int ldb, int *devInfo, int batch_count)

hipsolverStatus_t **hipsolverDnSpotrsBatched**(*hipsolverHandle_t* handle, *hipblasFillMode_t* uplo, int n, int nrhs, float *A[], int lda, float *B[], int ldb, int *devInfo, int batch_count)

hipsolverDn<type>getrs()

hipsolverStatus_t **hipsolverDnXgetrs**(*hipsolverDnHandle_t* handle, hipsolverDnParams_t params, *hipsolverOperation_t* trans, int64_t n, int64_t nrhs, hipDataType dataTypeA, const void *A, int64_t lda, const int64_t *devIpiv, hipDataType dataTypeB, void *B, int64_t ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnZgetrs**(*hipsolverHandle_t* handle, *hipblasOperation_t* trans, int n, int nrhs, const hipDoubleComplex *A, int lda, const int *devIpiv, hipDoubleComplex *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnCgetrs**(*hipsolverHandle_t* handle, *hipblasOperation_t* trans, int n, int nrhs, const hipFloatComplex *A, int lda, const int *devIpiv, hipFloatComplex *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnDgetrs**(*hipsolverHandle_t* handle, *hipblasOperation_t* trans, int n, int nrhs, const double *A, int lda, const int *devIpiv, double *B, int ldb, int *devInfo)

hipsolverStatus_t **hipsolverDnSgetrs**(*hipsolverHandle_t* handle, *hipblasOperation_t* trans, int n, int nrhs, const float *A, int lda, const int *devIpiv, float *B, int ldb, int *devInfo)

hipsolverDn<type><type>gesv_bufferSize()

hipsolverStatus_t **hipsolverDnZZgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnCCgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnDDgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, double *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnSSgesv_bufferSize**(*hipsolverHandle_t* handle, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, float *X, int ldx, void *work, size_t *lwork)

hipsolverDn<type><type>gesv()

hipsolverStatus_t **hipsolverDnZZgesv**(*hipsolverHandle_t* handle, int n, int nrhs, hipDoubleComplex *A, int lda, int *devIpiv, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnCCgesv**(*hipsolverHandle_t* handle, int n, int nrhs, hipFloatComplex *A, int lda, int *devIpiv, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnDDgesv**(*hipsolverHandle_t* handle, int n, int nrhs, double *A, int lda, int *devIpiv, double *B, int ldb, double *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnSSgesv**(*hipsolverHandle_t* handle, int n, int nrhs, float *A, int lda, int *devIpiv, float *B, int ldb, float *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

5.4.5 Least-squares solvers

List of least-squares solvers

- *hipsolverDn<type><type>gels_bufferSize()*
- *hipsolverDn<type><type>gels()*

hipsolverDn<type><type>gels_bufferSize()

hipsolverStatus_t **hipsolverDnZZgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnCCgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnDDgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, double *A, int lda, double *B, int ldb, double *X, int ldx, void *work, size_t *lwork)

hipsolverStatus_t **hipsolverDnSSgels_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int nrhs, float *A, int lda, float *B, int ldb, float *X, int ldx, void *work, size_t *lwork)

hipsolverDn<type><type>gels()

hipsolverStatus_t **hipsolverDnZZgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, hipDoubleComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnCCgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, hipFloatComplex *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnDDgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, double *A, int lda, double *B, int ldb, double *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

hipsolverStatus_t **hipsolverDnSSgels**(*hipsolverHandle_t* handle, int m, int n, int nrhs, float *A, int lda, float *B, int ldb, float *X, int ldx, void *work, size_t lwork, int *niters, int *devInfo)

5.4.6 Symmetric eigensolvers

List of symmetric eigensolvers

- *hipsolverDn<type>syevd_bufferSize()*
- *hipsolverDn<type>heevd_bufferSize()*
- *hipsolverDn<type>syevd()*
- *hipsolverDn<type>heevd()*
- *hipsolverDn<type>sygvd_bufferSize()*
- *hipsolverDn<type>hegvd_bufferSize()*
- *hipsolverDn<type>sygvd()*
- *hipsolverDn<type>hegvd()*

hipsolverDn<type>syevd_bufferSize()

hipsolverStatus_t **hipsolverDnDsyevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnSsyevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *W, int *lwork)

hipsolverDn<type>heevd_bufferSize()

hipsolverStatus_t **hipsolverDnZheevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnCheevd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const float *W, int *lwork)

hipsolverDn<type>syevd()

hipsolverStatus_t **hipsolverDnDsyevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsyevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *W, float *work, int lwork, int *devInfo)

hipsolverDn<type>heevd()

hipsolverStatus_t **hipsolverDnZheevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCheevd**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>sygvd_bufferSize()

hipsolverStatus_t **hipsolverDnDsygvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *B, int ldb, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnSsygvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *B, int ldb, const float *W, int *lwork)

hipsolverDn<type>hegvd_bufferSize()

hipsolverStatus_t **hipsolverDnZhegvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *B, int ldb, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnChegvd_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *B, int ldb, const float *W, int *lwork)

hipsolverDn<type>sygvd()

hipsolverStatus_t **hipsolverDnDsygvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsygvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, float *work, int lwork, int *devInfo)

hipsolverDn<type>hegvd()

hipsolverStatus_t **hipsolverDnZhegvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnChegvd**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, hipFloatComplex *work, int lwork, int *devInfo)

5.4.7 Singular value decomposition

List of SVD related functions

- *hipsolverDn<type>gesvd_bufferSize()*
- *hipsolverDn<type>gesvd()*

hipsolverDn<type>gesvd_bufferSize()

hipsolverStatus_t **hipsolverDnZgesvd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnCgesvd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnDgesvd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverStatus_t **hipsolverDnSgesvd_bufferSize**(*hipsolverHandle_t* handle, int m, int n, int *lwork)

hipsolverDn<type>gesvd()

hipsolverStatus_t **hipsolverDnZgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, double *rwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, float *rwork, int *devInfo)

hipsolverStatus_t **hipsolverDnDgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, double *rwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSgesvd**(*hipsolverHandle_t* handle, signed char jobu, signed char jobv, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, float *rwork, int *devInfo)

5.5 Dense matrix LAPACK-like functions

Other Lapack-like routines provided by hipSOLVER. These are divided into the following subcategories:

- *Symmetric eigensolvers*. Eigenproblems for symmetric matrices.
- *Singular value decomposition*. Singular values and related problems for general matrices.

5.5.1 Symmetric eigensolvers

List of Lapack-like symmetric eigensolvers

- *hipsolverDn<type>syevdx_bufferSize()*

hipsolverDn<type>syevdx_bufferSize()

hipsolverStatus_t **hipsolverDnDsyevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const double *A, int lda, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnSsyevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const float *A, int lda, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolverDn<type>heevdx_bufferSize()

hipsolverStatus_t **hipsolverDnZheevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnCheevdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolverDn<type>syevdx()

hipsolverStatus_t **hipsolverDnDsyevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, double *A, int lda, double vl, double vu, int il, int iu, int *nev, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsyevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, float *A, int lda, float vl, float vu, int il, int iu, int *nev, float *W, float *work, int lwork, int *devInfo)

hipsolverDn<type>heevdx()

hipsolverStatus_t **hipsolverDnZheevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double vl, double vu, int il, int iu, int *nev, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnCheevdx**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float vl, float vu, int il, int iu, int *nev, float *W, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>syevj_bufferSize()

hipsolverStatus_t **hipsolverDnDsyevj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnSsyevj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>heevj_bufferSize()

hipsolverStatus_t **hipsolverDnZheevj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnCheevj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>syevjBatched_bufferSize()

hipsolverStatus_t **hipsolverDnDsyevjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnSsyevjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverDn<type>heevjBatched_bufferSize()

hipsolverStatus_t **hipsolverDnZheevjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const double *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnCheevjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const float *W, int *lwork, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverDn<type>syevj()

hipsolverStatus_t **hipsolverDnDsyevj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnSsyevj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>heevj()

hipsolverStatus_t **hipsolverDnZheevj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnCheevj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>syevjBatched()

hipsolverStatus_t **hipsolverDnDsyevjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnSsyevjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverDn<type>heevjBatched()

hipsolverStatus_t **hipsolverDnZheevjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnCheevjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params, int batch_count)

hipsolverDn<type>sygvdx_bufferSize()

hipsolverStatus_t **hipsolverDnDsygvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *B, int ldb, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnSsygvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *B, int ldb, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolverDn<type>hegvdx_bufferSize()

hipsolverStatus_t **hipsolverDnZhegvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *B, int ldb, double vl, double vu, int il, int iu, int *nev, const double *W, int *lwork)

hipsolverStatus_t **hipsolverDnChegvdx_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *B, int ldb, float vl, float vu, int il, int iu, int *nev, const float *W, int *lwork)

hipsolverDn<type>sygvdx()

hipsolverStatus_t **hipsolverDnDsygvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double vl, double vu, int il, int iu, int *nev, double *W, double *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnSsygvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float vl, float vu, int il, int iu, int *nev, float *W, float *work, int lwork, int *devInfo)

hipsolverDn<type>hegvdx()

hipsolverStatus_t **hipsolverDnZhegvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double vl, double vu, int il, int iu, int *nev, double *W, hipDoubleComplex *work, int lwork, int *devInfo)

hipsolverStatus_t **hipsolverDnChegvdx**(*hipsolverHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipsolverEigRange_t* range, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float vl, float vu, int il, int iu, int *nev, float *W, hipFloatComplex *work, int lwork, int *devInfo)

hipsolverDn<type>sygvj_bufferSize()

hipsolverStatus_t **hipsolverDnDsygvj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const double *A, int lda, const double *B, int ldb, const double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnSsygvj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const float *A, int lda, const float *B, int ldb, const float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>hegvj_bufferSize()

hipsolverStatus_t **hipsolverDnZhegvj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipDoubleComplex *A, int lda, const hipDoubleComplex *B, int ldb, const double *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnChegvj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, const hipFloatComplex *A, int lda, const hipFloatComplex *B, int ldb, const float *W, int *lwork, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>sygvj()

hipsolverStatus_t **hipsolverDnDsygvj**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, double *A, int lda, double *B, int ldb, double *W, double *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnSsygvj**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, float *A, int lda, float *B, int ldb, float *W, float *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverDn<type>hegvj()

hipsolverStatus_t **hipsolverDnZhegvj**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipDoubleComplex *A, int lda, hipDoubleComplex *B, int ldb, double *W, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

hipsolverStatus_t **hipsolverDnChegvj**(*hipsolverDnHandle_t* handle, *hipsolverEigType_t* itype, *hipsolverEigMode_t* jobz, *hipblasFillMode_t* uplo, int n, hipFloatComplex *A, int lda, hipFloatComplex *B, int ldb, float *W, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverSyevjInfo_t* params)

5.5.2 Singular value decomposition

List of Lapack-like SVD related functions

- *hipsolverDn<type>gesvdj_bufferSize()*
- *hipsolverDn<type>gesvdjBatched_bufferSize()*
- *hipsolverDn<type>gesvdj()*
- *hipsolverDn<type>gesvdjBatched()*
- *hipsolverDn<type>gesvdaStridedBatched_bufferSize()*
- *hipsolverDn<type>gesvdaStridedBatched()*

hipsolverDn<type>gesvdj_bufferSize()

hipsolverStatus_t **hipsolverDnZgesvdj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const hipDoubleComplex *A, int lda, const double *S, const hipDoubleComplex *U, int ldu, const hipDoubleComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnCgesvdj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const hipFloatComplex *A, int lda, const float *S, const hipFloatComplex *U, int ldu, const hipFloatComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnDgesvdj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const double *A, int lda, const double *S, const double *U, int ldu, const double *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnSgesvdj_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, const float *A, int lda, const float *S, const float *U, int ldu, const float *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params)

hipsolverDn<type>gesvdjBatched_bufferSize()

hipsolverStatus_t **hipsolverDnZgesvdjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const hipDoubleComplex *A, int lda, const double *S, const hipDoubleComplex *U, int ldu, const hipDoubleComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnCgesvdjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const hipFloatComplex *A, int lda, const float *S, const hipFloatComplex *U, int ldu, const hipFloatComplex *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnDgesvdjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const double *A, int lda, const double *S, const double *U, int ldu, const double *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnSgesvdjBatched_bufferSize**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, const float *A, int lda, const float *S, const float *U, int ldu, const float *V, int ldv, int *lwork, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverDn<type>gesvdj()

hipsolverStatus_t **hipsolverDnZgesvdj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnCgesvdj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnDgesvdj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverStatus_t **hipsolverDnSgesvdj**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int econ, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params)

hipsolverDn<type>gesvdjBatched()

hipsolverStatus_t **hipsolverDnZgesvdjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, hipDoubleComplex *A, int lda, double *S, hipDoubleComplex *U, int ldu, hipDoubleComplex *V, int ldv, hipDoubleComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnCgesvdjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, hipFloatComplex *A, int lda, float *S, hipFloatComplex *U, int ldu, hipFloatComplex *V, int ldv, hipFloatComplex *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnDgesvdjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, double *A, int lda, double *S, double *U, int ldu, double *V, int ldv, double *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverStatus_t **hipsolverDnSgesvdjBatched**(*hipsolverDnHandle_t* handle, *hipsolverEigMode_t* jobz, int m, int n, float *A, int lda, float *S, float *U, int ldu, float *V, int ldv, float *work, int lwork, int *devInfo, *hipsolverGesvdjInfo_t* params, int batch_count)

hipsolverDn<type>gesvdaStridedBatched_bufferSize()

hipsolverStatus_t **hipsolverDnZgesvdaStridedBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const hipDoubleComplex *A, int lda, long long int strideA, const double *S, long long int strideS, const hipDoubleComplex *U, int ldu, long long int strideU, const hipDoubleComplex *V, int ldv, long long int strideV, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverDnCgesvdaStridedBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const hipFloatComplex *A, int lda, long long int strideA, const float *S, long long int strideS, const hipFloatComplex *U, int ldu, long long int strideU, const hipFloatComplex *V, int ldv, long long int strideV, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverDnDgesvdaStridedBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const double *A, int lda, long long int strideA, const double *S, long long int strideS, const double *U, int ldu, long long int strideU, const double *V, int ldv, long long int strideV, int *lwork, int batch_count)

hipsolverStatus_t **hipsolverDnSgesvdaStridedBatched_bufferSize**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const float *A, int lda, long long int strideA, const float *S, long long int strideS, const float *U, int ldu, long long int strideU, const float *V, int ldv, long long int strideV, int *lwork, int batch_count)

hipsolverDn<type>gesvdaStridedBatched()

hipsolverStatus_t **hipsolverDnZgesvdaStridedBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const hipDoubleComplex *A, int lda, long long int strideA, double *S, long long int strideS, hipDoubleComplex *U, int ldu, long long int strideU, hipDoubleComplex *V, int ldv, long long int strideV, hipDoubleComplex *work, int lwork, int *devInfo, double *hRnrmF, int batch_count)

hipsolverStatus_t **hipsolverDnCgesvdaStridedBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const hipFloatComplex *A, int lda, long long int strideA, float *S, long long int strideS, hipFloatComplex *U, int ldu, long long int strideU, hipFloatComplex *V, int ldv, long long int strideV, hipFloatComplex *work, int lwork, int *devInfo, double *hRnrmF, int batch_count)

hipsolverStatus_t **hipsolverDnDgesvdaStridedBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const double *A, int lda, long long int strideA, double *S, long long int strideS, double *U, int ldu, long long int strideU, double *V, int ldv, long long int strideV, double *work, int lwork, int *devInfo, double *hRnrmF, int batch_count)

hipsolverStatus_t **hipsolverDnSgesvdaStridedBatched**(*hipsolverHandle_t* handle, *hipsolverEigMode_t* jobz, int rank, int m, int n, const float *A, int lda, long long int strideA, float *S, long long int strideS, float *U, int ldu, long long int strideU, float *V, int ldv, long long int strideV, float *work, int lwork, int *devInfo, double *hRnrmF, int batch_count)

CHAPTER

SIX

HIPSOLVER COMPATIBILITY API - SPARSE MATRICES

This document provides the method signatures for the wrapper functions that are currently implemented in hipSOLVER. For a complete description of the functions' behavior and arguments, see the corresponding backend documentation at cuSOLVER API and/or rocSOLVER API.

For ease of porting from existing cuSOLVER applications to hipSOLVER, functions in the hipsolverSp compatibility API are designed to have method signatures that are consistent with the cusolverSp interface. At present, equivalent functions have not been added to hipSOLVER's regular API. Note that there are *some performance limitations* when using the rocSOLVER backend as not all the functionality required for optimal performance has been implemented yet.

6.1 Sparse matrix datatypes

hipSOLVER defines types and enumerations that are internally converted to the corresponding backend types at runtime. Here we list the types used in this compatibility API.

6.1.1 hipSOLVER compatibility API types

hipsolverSpHandle_t

typedef void *`hipsolverSpHandle_t`

hipsparseMatDescr_t

typedef void *`hipsparseMatDescr_t`

hipsolverStatus_t

See *hipsolverStatus_t*.

6.2 Sparse matrix helper functions

These are helper functions that control aspects of the hipSOLVER library. They are divided into the following categories:

- *Handle set-up and tear-down* functions. Used to initialize and cleanup the library handle.
- *Stream manipulation* functions. Provide functionality to manipulate streams.

6.2.1 Handle set-up and tear-down

List of handle initialization functions

- *hipsolverSpCreate()*
- *hipsolverSpDestroy()*

hipsolverSpCreate()

hipsolverStatus_t **hipsolverSpCreate**(*hipsolverSpHandle_t* *handle)

hipsolverSpDestroy()

hipsolverStatus_t **hipsolverSpDestroy**(*hipsolverSpHandle_t* handle)

6.2.2 Stream manipulation

List of stream manipulation functions

- *hipsolverSpSetStream()*

hipsolverSpSetStream()

hipsolverStatus_t **hipsolverSpSetStream**(*hipsolverSpHandle_t* handle, hipStream_t streamId)

6.3 Sparse matrix functions

Sparse matrix routines to solve complex Numerical Linear Algebra problems for sparse matrices. These functions are organized in the following categories:

6.3.1 Combined factorization and linear-system solvers

List of combined factorization and linear-system solvers

- *hipsolverSp<type>csrlsvchol()*
- *hipsolverSp<type>csrlsvcholHost()*

hipsolverSp<type>csrlsvchol()

hipsolverStatus_t **hipsolverSpDcsrlsvchol**(*hipsolverSpHandle_t* handle, int n, int nnzA, const *hipsparseMatDescr_t* descrA, const double *csrVal, const int *csrRowPtr, const int *csrColInd, const double *b, double tolerance, int reorder, double *x, int *singularity)

hipsolverStatus_t **hipsolverSpScsrlsvchol**(*hipsolverSpHandle_t* handle, int n, int nnzA, const *hipsparseMatDescr_t* descrA, const float *csrVal, const int *csrRowPtr, const int *csrColInd, const float *b, float tolerance, int reorder, float *x, int *singularity)

hipsolverSp<type>csrlsvcholHost()

hipsolverStatus_t **hipsolverSpDcsrlsvcholHost**(*hipsolverSpHandle_t* handle, int n, int nnzA, const *hipsparseMatDescr_t* descrA, const double *csrVal, const int *csrRowPtr, const int *csrColInd, const double *b, double tolerance, int reorder, double *x, int *singularity)

hipsolverStatus_t **hipsolverSpScsrlsvcholHost**(*hipsolverSpHandle_t* handle, int n, int nnzA, const *hipsparseMatDescr_t* descrA, const float *csrVal, const int *csrRowPtr, const int *csrColInd, const float *b, float tolerance, int reorder, float *x, int *singularity)

CHAPTER

SEVEN

HIPSOLVER COMPATIBILITY API - REFACTORIZATION

This document provides the method signatures for the wrapper functions that are currently implemented in hipSOLVER. For a complete description of the functions' behavior and arguments, see the corresponding backend documentation at cuSOLVER API and/or rocSOLVER API.

For ease of porting from existing cuSOLVER applications to hipSOLVER, functions in the hipsolverRf compatibility API are designed to have method signatures that are consistent with the cusolverRf interface. At present, equivalent functions have not been added to hipSOLVER's regular API.

7.1 Refactorization datatypes

hipSOLVER defines types and enumerations that are internally converted to the corresponding backend types at runtime. Here we list the types used in the this compatibility API.

7.1.1 hipSOLVER compatibility API types

List of types in the compatibility API

- *hipsolverRfHandle_t*
- *hipsolverRfFactorization_t*
- *hipsolverRfMatrixFormat_t*
- *hipsolverRfNumericBoostReport_t*
- *hipsolverRfResetValuesFastMode_t*
- *hipsolverRfTriangularSolve_t*
- *hipsolverRfUnitDiagonal_t*
- *hipsolverStatus_t*

hipsolverRfHandle_t

typedef void *hipsolverRfHandle_t

hipsolverRfFactorization_t

enum hipsolverRfFactorization_t

Values:

enumerator HIPSOLVERRF_FACTORIZATION_ALG0

enumerator HIPSOLVERRF_FACTORIZATION_ALG1

enumerator HIPSOLVERRF_FACTORIZATION_ALG2

hipsolverRfMatrixFormat_t

enum hipsolverRfMatrixFormat_t

Values:

enumerator HIPSOLVERRF_MATRIX_FORMAT_CSR

enumerator HIPSOLVERRF_MATRIX_FORMAT_CSC

hipsolverRfNumericBoostReport_t

enum hipsolverRfNumericBoostReport_t

Values:

enumerator HIPSOLVERRF_NUMERIC_BOOST_NOT_USED

enumerator HIPSOLVERRF_NUMERIC_BOOST_USED

hipsolverRfResetValuesFastMode_t

enum hipsolverRfResetValuesFastMode_t

Values:

enumerator HIPSOLVERRF_RESET_VALUES_FAST_MODE_OFF

enumerator HIPSOLVERRF_RESET_VALUES_FAST_MODE_ON

hipsolverRfTriangularSolve_t

enum **hipsolverRfTriangularSolve_t**

Values:

enumerator **HIPSOLVERRF_TRIANGULAR_SOLVE_ALG1**

enumerator **HIPSOLVERRF_TRIANGULAR_SOLVE_ALG2**

enumerator **HIPSOLVERRF_TRIANGULAR_SOLVE_ALG3**

hipsolverRfUnitDiagonal_t

enum **hipsolverRfUnitDiagonal_t**

Values:

enumerator **HIPSOLVERRF_UNIT_DIAGONAL_STORED_L**

enumerator **HIPSOLVERRF_UNIT_DIAGONAL_STORED_U**

enumerator **HIPSOLVERRF_UNIT_DIAGONAL_ASSUMED_L**

enumerator **HIPSOLVERRF_UNIT_DIAGONAL_ASSUMED_U**

hipsolverStatus_t

See *hipsolverStatus_t*.

7.2 Refactorization helper functions

These are helper functions that control aspects of the hipSOLVER library. They are divided into the following categories:

- *Handle set-up and tear-down* functions. Used to initialize and cleanup the library handle.
- *Input manipulation* functions. Provide functionality to manipulate function input.
- *Output manipulation* functions. Provide functionality to access function output.
- *Parameter manipulation* functions. Provide functionality to manipulate parameters.

7.2.1 Handle set-up and tear-down

List of handle initialization functions

- *hipsolverRfCreate()*
- *hipsolverRfDestroy()*

hipsolverRfCreate()

hipsolverStatus_t **hipsolverRfCreate**(*hipsolverRfHandle_t* *handle)

hipsolverRfDestroy()

hipsolverStatus_t **hipsolverRfDestroy**(*hipsolverRfHandle_t* handle)

7.2.2 Input manipulation

List of input functions

- *hipsolverRfSetupDevice()*
- *hipsolverRfSetupHost()*
- *hipsolverRfBatchSetupHost()*
- *hipsolverRfAnalyze()*
- *hipsolverRfBatchAnalyze()*
- *hipsolverRfResetValues()*
- *hipsolverRfBatchResetValues()*

hipsolverRfSetupDevice()

hipsolverStatus_t **hipsolverRfSetupDevice**(int n, int nnzA, int *csrRowPtrA, int *csrColIndA, double *csrValA, int nnzL, int *csrRowPtrL, int *csrColIndL, double *csrValL, int nnzU, int *csrRowPtrU, int *csrColIndU, double *csrValU, int *P, int *Q, *hipsolverRfHandle_t* handle)

hipsolverRfSetupHost()

hipsolverStatus_t **hipsolverRfSetupHost**(int n, int nnzA, int *h_csrRowPtrA, int *h_csrColIndA, double *h_csrValA, int nnzL, int *h_csrRowPtrL, int *h_csrColIndL, double *h_csrValL, int nnzU, int *h_csrRowPtrU, int *h_csrColIndU, double *h_csrValU, int *h_P, int *h_Q, *hipsolverRfHandle_t* handle)

hipsolverRfBatchSetupHost()

hipsolverStatus_t **hipsolverRfBatchSetupHost**(int batchSize, int n, int nnzA, int *h_csrRowPtrA, int *h_csrColIndA, double *h_csrValA_array[], int nnzL, int *h_csrRowPtrL, int *h_csrColIndL, double *h_csrValL, int nnzU, int *h_csrRowPtrU, int *h_csrColIndU, double *h_csrValU, int *h_P, int *h_Q, *hipsolverRfHandle_t* handle)

hipsolverRfAnalyze()

hipsolverStatus_t **hipsolverRfAnalyze**(*hipsolverRfHandle_t* handle)

hipsolverRfBatchAnalyze()

hipsolverStatus_t **hipsolverRfBatchAnalyze**(*hipsolverRfHandle_t* handle)

hipsolverRfResetValues()

hipsolverStatus_t **hipsolverRfResetValues**(int n, int nnzA, int *csrRowPtrA, int *csrColIndA, double *csrValA, int *P, int *Q, *hipsolverRfHandle_t* handle)

hipsolverRfBatchResetValues()

hipsolverStatus_t **hipsolverRfBatchResetValues**(int batchSize, int n, int nnzA, int *csrRowPtrA, int *csrColIndA, double *csrValA_array[], int *P, int *Q, *hipsolverRfHandle_t* handle)

7.2.3 Output manipulation

List of output functions

- *hipsolverRfAccessBundledFactorsDevice()*
- *hipsolverRfExtractBundledFactorsHost()*
- *hipsolverRfExtractSplitFactorsHost()*
- *hipsolverRfBatchZeroPivot()*

hipsolverRfAccessBundledFactorsDevice()

hipsolverStatus_t **hipsolverRfAccessBundledFactorsDevice**(*hipsolverRfHandle_t* handle, int *nnzM, int **Mp, int **Mi, double **Mx)

hipsolverRfExtractBundledFactorsHost()

hipsolverStatus_t **hipsolverRfExtractBundledFactorsHost**(*hipsolverRfHandle_t* handle, int *h_nnzM, int **h_Mp, int **h_Mi, double **h_Mx)

hipsolverRfExtractSplitFactorsHost()

hipsolverStatus_t **hipsolverRfExtractSplitFactorsHost**(*hipsolverRfHandle_t* handle, int *h_nnzL, int **h_Lp, int **h_Li, double **h_Lx, int *h_nnzU, int **h_Up, int **h_Ui, double **h_Ux)

hipsolverRfBatchZeroPivot()

hipsolverStatus_t **hipsolverRfBatchZeroPivot**(*hipsolverRfHandle_t* handle, int *position)

7.2.4 Parameter manipulation

List of parameter functions

hipsolverRfGet_Algs()

hipsolverStatus_t **hipsolverRfGet_Algs**(*hipsolverRfHandle_t* handle, *hipsolverRfFactorization_t* *fact_alg, *hipsolverRfTriangularSolve_t* *solve_alg)

hipsolverRfGetMatrixFormat()

hipsolverStatus_t **hipsolverRfGetMatrixFormat**(*hipsolverRfHandle_t* handle, *hipsolverRfMatrixFormat_t* *format, *hipsolverRfUnitDiagonal_t* *diag)

hipsolverRfGetNumericBoostReport()

hipsolverStatus_t **hipsolverRfGetNumericBoostReport**(*hipsolverRfHandle_t* handle, *hipsolverRfNumericBoostReport_t* *report)

hipsolverRfGetNumericProperties()

hipsolverStatus_t **hipsolverRfGetNumericProperties**(*hipsolverRfHandle_t* handle, double *zero, double *boost)

hipsolverRfGetResetValuesFastMode()

hipsolverStatus_t **hipsolverRfGetResetValuesFastMode**(*hipsolverRfHandle_t* handle, *hipsolverRfResetValuesFastMode_t* *fastMode)

hipsolverRfSetAlgs()

hipsolverStatus_t **hipsolverRfSetAlgs**(*hipsolverRfHandle_t* handle, *hipsolverRfFactorization_t* fact_alg, *hipsolverRfTriangularSolve_t* solve_alg)

hipsolverRfSetMatrixFormat()

hipsolverStatus_t **hipsolverRfSetMatrixFormat**(*hipsolverRfHandle_t* handle, *hipsolverRfMatrixFormat_t* format, *hipsolverRfUnitDiagonal_t* diag)

hipsolverRfSetNumericProperties()

hipsolverStatus_t **hipsolverRfSetNumericProperties**(*hipsolverRfHandle_t* handle, double effective_zero, double boost_val)

hipsolverRfSetResetValuesFastMode()

hipsolverStatus_t **hipsolverRfSetResetValuesFastMode**(*hipsolverRfHandle_t* handle, *hipsolverRfResetValuesFastMode_t* fastMode)

7.3 Refactorization Functions

Refactoring routines to solve complex Numerical Linear Algebra problems for sparse matrices. These functions are organized in the following categories:

- *Triangular factorizations*.
- *Linear-systems solvers*. Based on triangular factorizations.

7.3.1 Triangular factorizations

List of triangular factorizations

- *hipsolverRfRefactor()*
- *hipsolverRfBatchRefactor()*

hipsolverRfRefactor()

hipsolverStatus_t **hipsolverRfRefactor**(*hipsolverRfHandle_t* handle)

hipsolverRfBatchRefactor()

hipsolverStatus_t **hipsolverRfBatchRefactor**(*hipsolverRfHandle_t* handle)

7.3.2 Linear-systems solvers

List of linear solvers

- *hipsolverRfSolve()*
- *hipsolverRfBatchSolve()*

hipsolverRfSolve()

hipsolverStatus_t **hipsolverRfSolve**(*hipsolverRfHandle_t* handle, int *P, int *Q, int nrhs, double *Temp, int ldt, double *XF, int ldxf)

hipsolverRfBatchSolve()

hipsolverStatus_t **hipsolverRfBatchSolve**(*hipsolverRfHandle_t* handle, int *P, int *Q, int nrhs, double *Temp, int ldt, double *XF_array[], int ldxf)

CHAPTER

EIGHT

LICENSE

INDEX

H