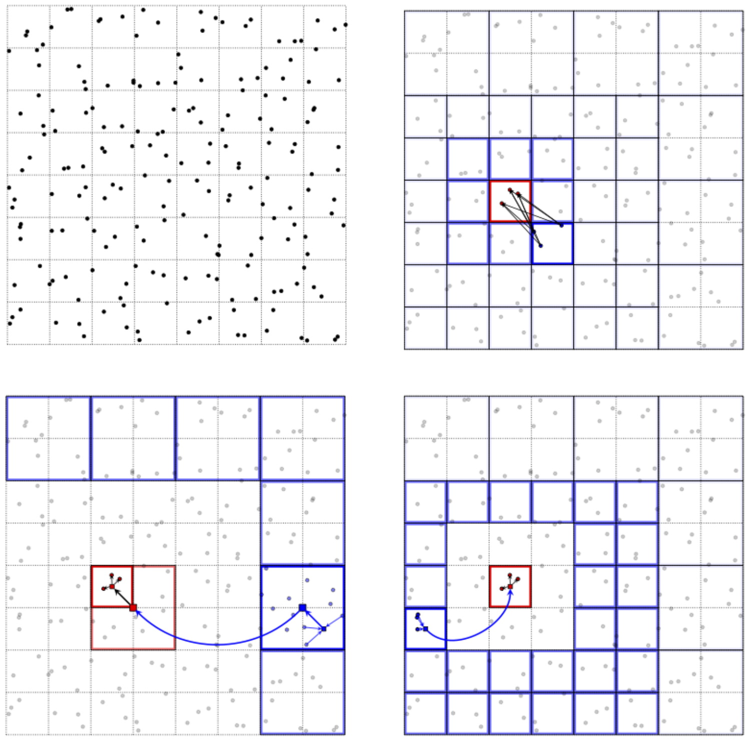
fmm3d Documentation

Release 1.0.0

Travis Askham Zydrunas Gimbutas
Leslie Greengard Libin Lu Jeremy Magland
Dhairya Malhotra Mike O'Neil Manas Rachh
Vladimir Rokhlin Felipe Vico

Nov 21, 2024

CONTENTS



FMM3D is a set of libraries to compute N-body interactions governed by the Laplace and Helmholtz equations, to a specified precision, in three dimensions, on a multi-core shared-memory machine. The library is written in Fortran, and has wrappers for C, MATLAB, and Python. As an example, given M arbitrary points $y_j \in \mathbb{R}^3$ with corresponding real numbers c_j, and N arbitrary points $x_j \in \mathbb{R}^3$, the Laplace FMM evaluates the N real numbers

$$u_\ell = \sum_{j=1}^{M} \frac{c_j}{4\pi \|x_\ell - y_j\|} , \qquad \text{for } \ell = 1, 2, \ldots N . \tag{1}$$

The y_j can be interpreted as source locations, c_j as charge strengths, and u_ℓ as the resulting potential at target location x_ℓ.

Such N-body interactions are needed in many applications in science and engineering, including molecular dynamics, astrophysics, rheology, and the numerical solution of partial differential equations. The naive CPU effort to evaluate (1) is $O(NM)$. The FMM approximates (1) to a requested relative precision ϵ with linear effort $O((M+N)\log^{3/2}(1/\epsilon))$.

The FMM relies on compressing the interactions between well-separated clusters of source and target points at a hierarchy of scales using analytic outgoing, incoming, and plane-wave expansions of the interaction kernel and associated translation operators. This library is an improved version of the FMMLIB3D software, Copyright (C) 2010-2012: Leslie Greengard and Zydrunas Gimbutas, released under the BSD license. **We provide two implementations of the library - an easy to install version with minimal dependencies, and a high-performance optimized version (which on some CPUs is 3x faster than the "easy" version).** For detailed instructions, see installation. The major improvements are the following:

- The use of plane wave expansions for diagonalizing the outgoing to incoming translation operators
- Vectorization of the FMM, to apply the same kernel with the same source and target locations to multiple strength vectors
- Optimized direct evaluation of the kernels using the SCTL library
- A redesign of the adaptive tree data structure

For sources and targets distributed in the volume, this code is about 4 times faster than the previous generation on a single CPU core, and for sources and targets distributed on a surface, this code is about 2 times faster.

Note

The present version of the code does not incorporate high frequency versions of the FMM. The plane wave expansions are used throughout for the Laplace FMM and for the Helmholtz case in the low frequency regime (for box sizes up to 32 wavelengths). At higher frequencies, the Helmholtz FMM uses the same `point and shoot` translation operators as FMMLIB3D, which results in sub-optimal performance.

Note

For very small repeated problems (less than 1000 input and output points), users should also consider dense matrix-matrix multiplication using BLAS3 (eg DGEMM,ZGEMM).

CHAPTER

ONE

INSTALLATION

1.1 Obtaining FMM3D

The source code can be downloaded from https://github.com/flatironinstitute/FMM3D

1.2 Dependencies

This library is supported for unix/linux, Mac OSX, and Windows.

For the basic libraries

- Fortran compiler, such as `gfortran` packaged with GCC
- GNU make

Optional:

- for building Python wrappers you will need `python3`, `pip3` and `numpy`.
- for building standard MATLAB wrappers: MATLAB
- for modifying MATLAB wrappers (experts only): `mwrap`

1.3 Quick install instructions

Make sure you have dependencies installed, and *cd* into your FMM3D directory.

- For linux, run `make install`.
- For Mac OSX, run `cp make.inc.macos.gnu make.inc` followed by `make install`.
- For Windows, run `cp make.inc.windows.mingw make.inc` followed by `make install`

This should compile the static library in `lib-static/`, the dynamic library in `lib/` and copy the dynamic library to `$(HOME)/lib` on Linux, to `/usr/local/lib` on Mac OSX, and to `C:\lib` on Windows. The location of the default installation directory can be changed by running:

```
make install PREFIX=(INSTALL_DIR)
```

In order to link against the dynamic library, you will have to update the `PATH` environment variable on Windows, `LD_LIBRARY_PATH` environment variable on Linux and `DYLD_LIBRARY_PATH` environment variable on Mac OSX to the installation directory. You may then link to the FMM library using the `-lfmm3d` option.

Note

On MacOSX, /usr/local/lib is included by default in the DYLD_LIBRARY_PATH.

To verify successful compilation of the program, run `make test` which compiles some fortran test drivers in `test/` linked against the static library, after which it runs the test programs. The last 14 lines of the terminal output should be:

```
cat print_testreshelm.txt
Successfully completed 5 out of 5 tests in helmrouts3d testing suite
Successfully completed 18 out of 18 tests in hfmm3d testing suite
Successfully completed 6 out of 6 tests in hfmm3d scale testing suite
Successfully completed 18 out of 18 tests in hfmm3d vec testing suite
Successfully completed 1 out of 1 tests in helm3d_mps testing suite
cat print_testreslap.txt
Successfully completed 5 out of 5 tests in laprouts3d testing suite
Successfully completed 18 out of 18 tests in lfmm3d testing suite
Successfully completed 2 out of 2 tests in lfmm3d scale testing suite
Successfully completed 18 out of 18 tests in lfmm3d vec testing suite
rm print_testreshelm.txt
rm print_testreslap.txt
```

To verify successful installation of the program, and the correct setting for environment variables, run `make test-dyn` which compiles some fortran test drivers in `test/` linked against the dynamic library, after which it runs teh test prgram. The output ofshould be the same as above.

Note

By default, `make install` creates the easy-to-install version of the library. To compile the library in its high-performance mode, append `FAST_KER=ON` to the make task. For instance `make install` should be replaced by `make install FAST_KER=ON`. See *Custom library compilation options* for other options.

If `make test` fails, see more detailed instructions below.

If `make test-dyn` fails with an error about not finding `-llibfmm3d_dll` or `-lfmm3d` or `libfmm3d` make sure that the appropriate environment variables have been set. If it fails with other issues, see more detailed instructions below.

Type `make` to see a list of other build options (language interfaces, etc). Please see Fortran and C interfaces and look in `examples/` for sample drivers.

If there is an error in testing on a standard set-up, please file a bug report as a New Issue at https://github.com/flatironinstitute/FMM3D/issues

1.3.1 Custom library compilation options

In the (default) easy-to-install version, the library is compiled without using the optimized direct evaluation kernels.

In order to disable multi-threading, append `OMP=OFF` to the make task.

In order to use the optimized direct evaluation kernels (this automatically turns on multithreading as well), append `FAST_KER=ON` to the make task. The optimized direct kernel evaluation rotuines require gcc version 9 or higher. This option is currently *not* supported on Windows.

All of these different libraries are built with the same name, so you will have to move them to other locations, or build a 2nd copy of the repo, if you want to keep both versions.

You *must* do at least `make objclean` before changing to the openmp /fast direct kernel evaluation options.

1.3.2 Examples

- `make examples` to compile and run the examples for calling from Fortran.
- `make c-examples` to compile and run the examples for calling from C.

The `examples` directory is a good place to see usage examples for Fortran. There are three sample Fortran drivers for both the Laplace and Helmholtz FMMs, one which demonstrates the use of FMMs, one which demonstrates the use of vectorized FMMs, and one which demonstrates the use of the same calling sequence as FMMLIB3D - so that legacy codes are backward compatible with FMMLIB3D.

The sample drivers for the Laplace FMM are `lfmm3d_example.f`, `lfmm3d_vec_example.f`, and `lfmm3d_legacy_example.f`, and the corresponding makefiles are `lfmm3d_example.make`, `lfmm3d_vec_example.make`, and `lfmm3d_legacy_example.make`. These demonstrate how to link to the dynamic library `libfmm3d.so`. The analogous Helmholtz drivers are `hfmm3d_example.f`, `hfmm3d_vec_example.f`, and `hfmm3d_legacy_example.f`. The corresponding makefiles are `hfmm3d_example.make`, `hfmm3d_vec_example.make`, and `hfmm3d_legacy_example.make`.

Analogous C sample drivers can be found in `c/`.

1.4 Building Python wrappers

First make sure you have python (version 3 or higher), pip and numpy installed.

You may then execute `make python` (after copying over the operating system specific make.inc.* file to make.inc) which calls pip for the install and then runs some tests.

To rerun the tests, you may run `pytest` in `python/` or alternatively run `python python/test_hfmm.py` and `python python/test_lfmm.py`.

See `python/hfmmexample.py` and `python/lfmmexample.py` to see usage examples for the Python wrappers.

Note

On windows, you will need to update `distutils.cfg` located in `(PYTHON_INSTALL_DIR)\Lib\distutils` and set it to:

```
[build]
compiler=mingw32

[build_ext]
compiler=mingw32
```

which forces python to use the mingw compiler for building its modules. In case you wish to revert to using VC/C++ for building python modules, make sure to update distutils.cfg appropriately.

1.4.1 A few words about Python environments

There can be confusion and conflicts between various versions of Python and installed packages. It is therefore a very good idea to use virtual environments. Here's a simple way to do it (after installing python-virtualenv):

```
Open a terminal
virtualenv -p /usr/bin/python3 env1
. env1/bin/activate
```

Now you are in a virtual environment that starts from scratch. All pip installed packages will go inside the env1 directory. (You can get out of the environment by typing `deactivate`)

It's advisable to install numpy into a virtual environment using pip as.

virtualenv -p /usr/bin/python3 env1 . env1/bin/activate pip install numpy

1.5 Building the MATLAB wrappers

First make sure you have MATLAB installed.

Then run `make matlab` (after copying over the operating system specific make.inc.* file to make.inc) which links the .m files to the .c file in the matlab folder.

To run tests, you can run `matlab test_hfmm3d.m` and `matlab test_lfmm3d.m` and it should return with 0 crashes.

Example codes for demonstrating the Helmholtz and Laplace interfaces are `hfmm3d_example.m` and `lfmm3d_example.m`.

In order to build the MATLAB routines with the optimized direct kernel evaluation routines on a Mac, we recommend building mex with gcc instead of clang. The relevant xml files for configuring mex to use gcc can be found at https://github.com/danfortunato/matlab-gcc. Follow the instructions there to configure mex with gcc, and set CC = `gcc-<version number>` in your make.inc file.

1.6 Tips for installing dependencies

1.6.1 On Ubuntu linux

On Ubuntu linux (assuming python3 as opposed to python):

```
sudo apt-get install make build-essential gfortran
```

1.6.2 On Fedora/CentOS linux

On a Fedora/CentOS linux system, these dependencies can be installed as follows:

```
sudo yum install make gcc gcc-c++ gcc-gfortran libgomp
```

1.6.3 On Mac OSX

First setup Homebrew as follows. If you don't have Xcode, install Command Line Tools by opening a terminal (from /Applications/Utilities/) and typing:

```
xcode-select --install
```

Then install Homebrew by pasting the installation command from https://brew.sh

Then do:

```
brew install gcc
```

1.6.4 On Windows

Download 64 bit mingw (Available at https://sourceforge.net/projects/mingw-w64/files/mingw-w64/mingw-w64-release/). Recommended version is `gcc-8.1.0`, `x86_64-posix-seh`. Follow the install instructions and append to the environment variable `PATH` the location of the bin directory of your mingw installation.

Download and install make for windows (Available at http://gnuwin32.sourceforge.net/packages/make.htm).

Download and install git for windows (Available at https://git-scm.com/download/win).

1.7 Tips for installing optional dependencies

1.7.1 Installing python and pip

On Ubuntu linux

```
sudo apt-get install python3 python3-pip
```

On Mac OSX

Make sure you have homebrew installed. See Tips for installing dependencies -> On Mac OSX

```
brew install python3
```

On Windows

Download and install python3.7 from python.org.

1.7.2 Configuring MATLAB

On Windows

Update MINGW_LPATH in make.inc.windows.mingw to point to the appropriate installation directory (it should be the one within the gcc folder).

To setup mingw as the C compiler on MATLAB run configuremingw.p (which can be downloaded from here) and choose the mingw directory. To verify successful setup run mex -setup from matlab and it should be configured to compile with mingw.

1.7.3 Installing MWrap

If you make any changes to the fortran code, you will need to regenerate the .c files from the .mw files for which mwrap is required. This is not needed for most users. MWrap is a very useful MEX interface generator by Dave Bindel.

Make sure you have flex and bison installed. Download version 0.33.5 or later from https://github.com/zgimbutas/mwrap, un-tar the package, cd into it, then:

```
make
sudo cp mwrap /usr/local/bin/
```

CHAPTER

TWO

DEFINITIONS

Let $x_j \in \mathbb{R}^3$, $i = 1, 2, \ldots N$, denote a collection of source locations and let $t_i \in \mathbb{R}^3$ denote a collection of target locations.

2.1 Laplace FMM

Let $c_j \in \mathbb{R}$, $j = 1, 2, \ldots N$, denote a collection of charge strengths, $v_j \in \mathbb{R}^3$, $j = 1, 2, \ldots N$, denote a collection of dipole strengths.

The Laplace FMM computes the potential $u(x)$ and the its gradient $\nabla u(x)$ given by

$$u(x) = \sum_{j=1}^{N} \frac{c_j}{4\pi \|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right) , \tag{2.1}$$

at the source and target locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

2.2 Helmholtz FMM

Let $c_j \in \mathbb{C}$, $j = 1, 2, \ldots N$, denote a collection of charge strengths, $v_j \in \mathbb{C}^3$, $j = 1, 2, \ldots N$, denote a collection of dipole strengths. Let $k \in \mathbb{C}$ denote the wave number or the Helmholtz parameter.

The Helmholtz FMM computes the potential $u(x)$ and the its gradient $\nabla u(x)$ given by

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x - x_j\|}}{4\pi \|x - x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x - x_j\|}}{4\pi \|x - x_j\|} \right) , \tag{2.2}$$

at the source and target locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

2.3 Vectorized versions

The vectorized versions of the Laplace and Helmholtz FMM, computes repeated FMMs for new charge and dipole strengths located at the same source locations, where the potential and its gradient are evaluated at the same set of target locations.

For example, for the vectorized Laplace FMM, let $c_{\ell,j} \in \mathbb{R}$, $j = 1, 2, \ldots N$, $\ell = 1, 2, \ldots n_d$ denote a collection of n_d charge strengths, and let $v_{\ell,j} \in \mathbb{R}^3$ denote a collection of n_d dipole strengths. Then the vectorized Laplace FMM computes the potentials $u_\ell(x)$ and its gradients $\nabla u_\ell(x)$ defined by the formula

$$u_\ell(x) = \sum_{j=1}^{N} \frac{c_{\ell,j}}{4\pi \|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right) , \quad \ell = 1, 2, \ldots n_d \tag{2.3}$$

at the source and target locations.

Similarly, for the vectorized Helmholtz FMM, let $c_{\ell,j} \in \mathbb{C}$, $j = 1, 2, \ldots N$, $\ell = 1, 2, \ldots n_d$ denote a collection of n_d charge strengths, and let $v_{\ell,j} \in \mathbb{C}^3$ denote a collection of n_d dipole strengths. Then the vectorized Helmholtz FMM computes the potentials $u_\ell(x)$ and its gradients $\nabla u_\ell(x)$ defined by the formula

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right) , \quad \ell = 1, 2, \ldots n_d \tag{2.4}$$

at the source and target locations.

Note

In double precision arithmetic, two numbers which are within machine precision of each other cannot be distinguished. In order to account for this, suppose that the sources and targets are contained in a cube with side length L, then for all x such that $\|x - x_j\| \leq L\varepsilon_{\text{mach}}$, the term corresponding to x_j is dropped from the sum. Here $\varepsilon_{\text{mach}} = 2^{-52}$ is machine precision.

CHAPTER

THREE

FORTRAN AND C INTERFACES

3.1 Laplace FMM

The Laplace FMM evaluates the following potential and its gradient

$$u(x) = \sum_{j=1}^{N} \frac{c_j}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right) .$$

Here x_j are the source locations, c_j are the charge strengths and v_j are the dipole strengths, and the collection of x at which the potential and its gradient are evaluated are referred to as the evalution points.

There are 18 different Fortran wrappers for the Laplace FMM to account for collection of evaluation points (sources only, targets only, sources+targets), interaction kernel (charges only, dipoles only, charges + dipoles), output request (potential, potential+gradient).

For example, the subroutine to evaluate the potential and gradient, at a collection of targets t_i due to a collection of charges is:

```
lfmm3d_t_c_g
```

In general, the subroutine names take the following form:

```
lfmm3d_<eval-pts>_<int-ker>_<out>
```

- <eval-pts>: evaluation points. Collection of x where u and its gradient is to be evaluated
  - s: Evaluate u and its gradient at the source locations x_i
  - t: Evaluate u and its gradient at t_i, a collection of target locations specified by the user.
  - st: Evaluate u and its gradient at both source and target locations x_i and t_i.
- <int-ker>: kernel of interaction (charges/dipoles/both). The charge interactions are given by $c_j/4\pi\|x-x_j\|$, and the dipole interactions are given by $-v_j \cdot \nabla(1/4\pi\|x - x_j\|)$
  - c: charges
  - d: dipoles

  - cd: charges + dipoles
- <out>: Flag for evaluating potential or potential + gradient
  - p: on output only u is evaluated
  - g: on output both u and its gradient are evaluated
  - h: on output u, its gradient and its hessian are evaluated

These are all the single density routines. To get a vectorized version of any of the routines use:

```
<subroutine name>_vec
```

> **Note**
>
> For the vectorized subroutines, the charge strengths, dipole strengths, potentials, and gradients are interleaved as opposed to provided in a sequential manner. For example for three sets of charge strengths, they should be stored as $c_{1,1}, c_{2,1}, c_{3,1}, c_{1,2}, c_{2,2}, c_{3,2} \ldots c_{1,N}, c_{2,N}, c_{3,N}$.

Example drivers:

- `examples/lfmm3d_example.f`. The corresponding makefile is `examples/lfmm3d_example.make`
- `examples/lfmm3d_vec_example.f`. The corresponding makefile is `examples/lfmm3d_vec_example.make`

Back to top

3.1.1 List of interfaces

- Evaluation points: Sources
  - Interaction Type: Charges
    - Potential (*lfmm3d_s_c_p*)
    - Gradient (*lfmm3d_s_c_g*)
    - Hessian (*lfmm3d_s_c_h*)
  - Interaction Type: Dipoles
    - Potential (*lfmm3d_s_d_p*)
    - Gradient (*lfmm3d_s_d_g*)
    - Hessian (*lfmm3d_s_d_h*)
  - Interaction Type: Charges + Dipoles
    - Potential (*lfmm3d_s_cd_p*)
    - Gradient (*lfmm3d_s_cd_g*)
    - Hessian (*lfmm3d_s_cd_h*)
- Evaluation points: Targets
  - Interaction Type: Charges
    - Potential (*lfmm3d_t_c_p*)
    - Gradient (*lfmm3d_t_c_g*)

    * Hessian (*lfmm3d_t_c_h*)
  - Interaction Type: Dipoles
    * Potential (*lfmm3d_t_d_p*)
    * Gradient (*lfmm3d_t_d_g*)
    * Hessian (*lfmm3d_t_d_h*)
  - Interaction Type: Charges + Dipoles
    * Potential (*lfmm3d_t_cd_p*)
    * Gradient (*lfmm3d_t_cd_g*)
    * Hessian (*lfmm3d_t_cd_h*)
- Evaluation points: Sources + Targets
  - Interaction Type: Charges
    * Potential (*lfmm3d_st_c_p*)
    * Gradient (*lfmm3d_st_c_g*)
    * Hessian (*lfmm3d_st_c_h*)
  - Interaction Type: Dipoles
    * Potential (*lfmm3d_st_d_p*)
    * Gradient (*lfmm3d_st_d_g*)
    * Hessian (*lfmm3d_st_d_h*)
  - Interaction Type: Charges + Dipoles
    * Potential (*lfmm3d_st_cd_p*)
    * Gradient (*lfmm3d_st_cd_g*)
    * Hessian (*lfmm3d_st_cd_h*)

Back to top

lfmm3d_s_c_p

- Evaluation points: Sources
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine lfmm3d_s_c_p(eps,nsource,source,charge,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_c_p_vec(nd,eps,nsource,source,charge,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_c_g

- Evaluation points: Sources
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_s_c_g(eps,nsource,source,charge,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_c_g_vec(nd,eps,nsource,source,charge,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi \|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_c_h

- Evaluation points: Sources
- Interaction kernel: Charges
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_s_c_h(eps,nsource,source,charge,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Vectorized version:

```
subroutine lfmm3d_s_c_h_vec(nd,eps,nsource,source,charge,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_d_p

- Evaluation points: Sources
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_s_d_p(eps,nsource,source,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_d_p_vec(nd,eps,nsource,source,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_d_g

- Evaluation points: Sources
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_s_d_g(eps,nsource,source,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_d_g_vec(nd,eps,nsource,source,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_d_h

- Evaluation points: Sources
- Interaction kernel: Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_s_d_h(eps,nsource,source,dipvec,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$

- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_d_h_vec(nd,eps,nsource,source,dipvec,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_cd_p

- Evaluation points: Sources
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_s_cd_p(eps,nsource,source,charge,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_cd_p_vec(nd,eps,nsource,source,charge,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_cd_g

- Evaluation points: Sources
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_s_cd_g(eps,nsource,source,charge,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_cd_g_vec(nd,eps,nsource,source,charge,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_s_cd_h

- Evaluation points: Sources
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_s_cd_h(eps,nsource,source,charge,dipvec,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources

- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_s_cd_h_vec(nd,eps,nsource,source,charge,dipvec,pot,grad,hess,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_c_p

- Evaluation points: Targets
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine lfmm3d_t_c_p(eps,nsource,source,charge,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_c_p_vec(nd,eps,nsource,source,charge,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_c_g

- Evaluation points: Targets
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_t_c_g(eps,nsource,source,charge,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j

- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_c_g_vec(nd,eps,nsource,source,charge,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_c_h

- Evaluation points: Targets
- Interaction kernel: Charges
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_t_c_h(eps,nsource,source,charge,ntarg,targ,pottarg,gradtarg,hesstarg,
→ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_c_h_vec(nd,eps,nsource,source,charge,ntarg,targ,pottarg,gradtarg,
→hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_d_p

- Evaluation points: Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_t_d_p(eps,nsource,source,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources

- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_d_p_vec(nd,eps,nsource,source,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_d_g

- Evaluation points: Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_t_d_g(eps,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla \nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_d_g_vec(nd,eps,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_d_h

- Evaluation points: Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_t_d_h(eps,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,hesstarg,
↪ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_d_h_vec(nd,eps,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,
↪hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_cd_p

- Evaluation points: Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_t_cd_p(eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Vectorized version:

```
subroutine lfmm3d_t_cd_p_vec(nd,eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_cd_g

- Evaluation points: Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_t_cd_g(eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets

- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_cd_g_vec(nd,eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_t_cd_h

- Evaluation points: Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_t_cd_h(eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,gradtarg,
↪hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_t_cd_h_vec(nd,eps,nsource,source,charge,dipvec,ntarg,targ,pottarg,
↪gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_c_p

- Evaluation points: Sources and Targets
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine lfmm3d_st_c_p(eps,nsource,source,charge,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_c_p_vec(nd,eps,nsource,source,charge,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_c_g

- Evaluation points: Sources and Targets
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_st_c_g(eps,nsource,source,charge,pot,grad,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$

- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_c_g_vec(nd,eps,nsource,source,charge,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_c_h

- Evaluation points: Sources and Targets
- Interaction kernel: Charges
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_st_c_h(eps,nsource,source,charge,pot,grad,hess,ntarg,targ,pottarg,
↪gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla\nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_c_h_vec(nd,eps,nsource,source,charge,pot,grad,hess,ntarg,targ,
↪pottarg,gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_d_p

- Evaluation points: Sources and Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_st_d_p(eps,nsource,source,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_d_p_vec(nd,eps,nsource,source,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_d_g

- Evaluation points: Sources and Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_st_d_g(eps,nsource,source,dipvec,pot,grad,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$

- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_d_g_vec(nd,eps,nsource,source,dipvec,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_d_h

- Evaluation points: Sources and Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_st_d_h(eps,nsource,source,dipvec,pot,grad,hess,ntarg,targ,pottarg,
↪gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla \nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla \nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_d_h_vec(nd,eps,nsource,source,dipvec,pot,grad,hess,ntarg,targ,
↪pottarg,gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_cd_p

- Evaluation points: Sources and Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine lfmm3d_st_cd_p(eps,nsource,source,charge,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi \|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi \|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_cd_p_vec(nd,eps,nsource,source,charge,dipvec,pot,ntarg,targ,pottarg,
↪ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_cd_g

- Evaluation points: Sources and Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine lfmm3d_st_cd_g(eps,nsource,source,charge,dipvec,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$

- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_cd_g_vec(nd,eps,nsource,source,charge,dipvec,pot,grad,ntarg,targ,
↪pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

lfmm3d_st_cd_h

- Evaluation points: Sources and Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential, Gradient and Hessians

```
subroutine lfmm3d_st_cd_h(eps,nsource,source,charge,dipvec,pot,grad,hess,ntarg,targ,
↪pottarg,gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double precision(nsource)**
  Charge strengths, c_j
- **dipvec: double precision(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double precision(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double precision(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **hess: double precision(6,nsource)**
  Hessian at source locations, $\nabla\nabla u(x_j)$
- **pottarg: double precision(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double precision(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **hesstarg: double precision(6,ntarg)**
  Hessian at target locations, $\nabla\nabla u(t_i)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine lfmm3d_st_cd_h_vec(nd,eps,nsource,source,charge,dipvec,pot,grad,hess,ntarg,
→targ,pottarg,gradtarg,hesstarg,ier)
```

This subroutine evaluates the potential, its gradient and its hessian

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{1}{4\pi\|x - x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double precision(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double precision(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double precision(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double precision(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **hess: double precision(nd,6,nsource)**
  Gradient at source locations, $\nabla\nabla u_\ell(x_j)$
- **pottarg: double precision(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double precision(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **hesstarg: double precision(nd,3,ntarg)**
  Hessian at target locations, $\nabla\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Laplace FMM

Back to top

3.2 Helmholtz FMM

The Helmholtz FMM evaluates the following potential and its gradient

$$u(x) = \sum_{j=1}^{N} \frac{c_j e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right) .$$

Here x_j are the source locations, c_j are the charge strengths and v_j are the dipole strengths, and the collection of x at which the potential and its gradient are evaluated are referred to as the evalution points.

There are 18 different Fortran wrappers for the Helmholtz FMM to account for collection of evaluation points (sources only, targets only, sources+targets), interaction kernel (charges only, dipoles only, charges + dipoles), output request (potential, potential+gradient).

For example, the subroutine to evaluate the potential and gradient, at a collection of targets t_i due to a collection of charges is:

```
hfmm3d_t_c_g
```

In general, the subroutine names take the following form:

```
hfmm3d_<eval-pts>_<int-ker>_<out>
```

- <eval-pts>: evaluation points. Collection of x where u and its gradient is to be evaluated
  - s: Evaluate u and its gradient at the source locations x_i
  - t: Evaluate u and its gradient at t_i, a collection of target locations specified by the user.
  - st: Evaluate u and its gradient at both source and target locations x_i and t_i.
- <int-ker>: kernel of interaction (charges/dipoles/both). The charge interactions are given by $c_j/4\pi\|x-x_j\|$, and the dipole interactions are given by $-v_j \cdot \nabla(1/4\pi\|x-x_j\|)$
  - c: charges
  - d: dipoles
  - cd: charges + dipoles
- <out>: Flag for evaluating potential or potential + gradient
  - p: on output only u is evaluated
  - g: on output both u and its gradient are evaluated

These are all the single density routines. To get a vectorized version of any of the routines use:

```
<subroutine name>_vec
```

Note

For the vectorized subroutines, the charge strengths, dipole strengths, potentials, and gradients are interleaved as opposed to provided in a sequential manner. For example for three sets of charge strengths, they should be stored as $c_{1,1}, c_{2,1}, c_{3,1}, c_{1,2}, c_{2,2}, c_{3,2} \ldots c_{1,N}, c_{2,N}, c_{3,N}$.

Example drivers:

- `examples/hfmm3d_example.f`. The corresponding makefile is `examples/hfmm3d_example.make`

- `examples/hfmm3d_vec_example.f`. The corresponding makefile is `examples/hfmm3d_vec_example.make`

Back to top

3.2.1 List of interfaces

- Evaluation points: Sources
  - Interaction Type: Charges
    - Potential (*hfmm3d_s_c_p*)
    - Gradient (*hfmm3d_s_c_g*)
  - Interaction Type: Dipoles
    - Potential (*hfmm3d_s_d_p*)
    - Gradient (*hfmm3d_s_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*hfmm3d_s_cd_p*)
    - Gradient (*hfmm3d_s_cd_g*)
- Evaluation points: Targets
  - Interaction Type: Charges
    - Potential (*hfmm3d_t_c_p*)
    - Gradient (*hfmm3d_t_c_g*)
  - Interaction Type: Dipoles
    - Potential (*hfmm3d_t_d_p*)
    - Gradient (*hfmm3d_t_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*hfmm3d_t_cd_p*)
    - Gradient (*hfmm3d_t_cd_g*)
- Evaluation points: Sources + Targets
  - Interaction Type: Charges
    - Potential (*hfmm3d_st_c_p*)
    - Gradient (*hfmm3d_st_c_g*)
  - Interaction Type: Dipoles
    - Potential (*hfmm3d_st_d_p*)
    - Gradient (*hfmm3d_st_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*hfmm3d_st_cd_p*)
    - Gradient (*hfmm3d_st_cd_g*)

Back to top

hfmm3d_s_c_p

- Evaluation points: Sources
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine hfmm3d_s_c_p(eps,zk,nsource,source,charge,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_c_p_vec(nd,eps,zk,nsource,source,charge,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_s_c_g

- Evaluation points: Sources
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_s_c_g(eps,zk,nsource,source,charge,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_c_g_vec(nd,eps,zk,nsource,source,charge,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_s_d_p

- Evaluation points: Sources
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_s_d_p(eps,zk,nsource,source,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_d_p_vec(nd,eps,zk,nsource,source,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_s_d_g

- Evaluation points: Sources
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_s_d_g(eps,zk,nsource,source,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_d_g_vec(nd,eps,zk,nsource,source,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_s_cd_p

- Evaluation points: Sources
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_s_cd_p(eps,zk,nsource,source,charge,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j

- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_cd_p_vec(nd,eps,zk,nsource,source,charge,dipvec,pot,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_s_cd_g

- Evaluation points: Sources
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_s_cd_g(eps,zk,nsource,source,charge,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_s_cd_g_vec(nd,eps,zk,nsource,source,charge,dipvec,pot,grad,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source locations $x = x_j$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_c_p

- Evaluation points: Targets
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine hfmm3d_t_c_p(eps,zk,nsource,source,charge,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_t_c_p_vec(nd,eps,zk,nsource,source,charge,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_c_g

- Evaluation points: Targets
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_t_c_g(eps,zk,nsource,source,charge,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_t_c_g_vec(nd,eps,zk,nsource,source,charge,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_d_p

- Evaluation points: Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_t_d_p(eps,zk,nsource,source,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_t_d_p_vec(nd,eps,zk,nsource,source,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_d_g

- Evaluation points: Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_t_d_g(eps,zk,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources

- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_t_d_g_vec(nd,eps,zk,nsource,source,dipvec,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_cd_p

- Evaluation points: Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_t_cd_p(eps,zk,nsource,source,charge,dipvec,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Vectorized version:

```
subroutine hfmm3d_t_cd_p_vec(nd,eps,zk,nsource,source,charge,dipvec,ntarg,targ,pottarg,
↪ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_t_cd_g

- Evaluation points: Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_t_cd_g(eps,zk,nsource,source,charge,dipvec,ntarg,targ,pottarg,gradtarg,
↪ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources

- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_t_cd_g_vec(nd,eps,zk,nsource,source,charge,dipvec,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the target locations $x = t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_c_p

- Evaluation points: Sources and Targets
- Interaction kernel: Charges
- Outputs requested: Potential

```
subroutine hfmm3d_st_c_p(eps,zk,nsource,source,charge,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_st_c_p_vec(nd,eps,zk,nsource,source,charge,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_c_g

- Evaluation points: Sources and Targets
- Interaction kernel: Charges
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_st_c_g(eps,zk,nsource,source,charge,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested

- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_st_c_g_vec(nd,eps,zk,nsource,source,charge,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$

- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_d_p

- Evaluation points: Sources and Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_st_d_p(eps,zk,nsource,source,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$

- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_st_d_p_vec(nd,eps,zk,nsource,source,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_d_g

- Evaluation points: Sources and Targets
- Interaction kernel: Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_st_d_g(eps,zk,nsource,source,dipvec,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = -\sum_{j=1}^{N} v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_st_d_g_vec(nd,eps,zk,nsource,source,dipvec,pot,grad,ntarg,targ,pottarg,
↪gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = -\sum_{j=1}^{N} v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities

- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_cd_p

- Evaluation points: Sources and Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential

```
subroutine hfmm3d_st_cd_p(eps,zk,nsource,source,charge,dipvec,pot,ntarg,targ,pottarg,ier)
```

This subroutine evaluates the potential

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j

- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

---

Vectorized version:

```
subroutine hfmm3d_st_cd_p_vec(nd,eps,zk,nsource,source,charge,dipvec,pot,ntarg,targ,
↪pottarg,ier)
```

This subroutine evaluates the potential

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

hfmm3d_st_cd_g

- Evaluation points: Sources and Targets
- Interaction kernel: Charges and Dipoles
- Outputs requested: Potential and Gradient

```
subroutine hfmm3d_st_cd_g(eps,zk,nsource,source,charge,dipvec,pot,grad,ntarg,targ,
↪pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **eps: double precision**
  precision requested
- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **charge: double complex(nsource)**
  Charge strengths, c_j
- **dipvec: double complex(3,nsource)**
  Dipole strengths, v_j
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Target locations, t_i

Output arguments:

- **pot: double complex(nsource)**
  Potential at source locations, $u(x_j)$
- **grad: double complex(3,nsource)**
  Gradient at source locations, $\nabla u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, $u(t_i)$
- **gradtarg: double complex(3,ntarg)**
  Gradient at target locations, $\nabla u(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Vectorized version:

```
subroutine hfmm3d_st_cd_g_vec(nd,eps,zk,nsource,source,charge,dipvec,pot,grad,ntarg,targ,
↪pottarg,gradtarg,ier)
```

This subroutine evaluates the potential and its gradient

$$u_\ell(x) = \sum_{j=1}^{N} c_{\ell,j} \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_{\ell,j} \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

at the source and target locations $x = x_j, t_i$. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

Input arguments:

- **nd: integer**
  number of densities
- **charge: double complex(nd,nsource)**
  Charge strengths, $c_{\ell,j}$
- **dipvec: double complex(nd,3,nsource)**
  Dipole strengths, $v_{\ell,j}$

Output arguments:

- **pot: double complex(nd,nsource)**
  Potential at source locations, $u_\ell(x_j)$
- **grad: double complex(nd,3,nsource)**
  Gradient at source locations, $\nabla u_\ell(x_j)$
- **pottarg: double complex(nd,ntarg)**
  Potential at target locations, $u_\ell(t_i)$
- **gradtarg: double complex(nd,3,ntarg)**
  Gradient at target locations, $\nabla u_\ell(t_i)$
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Back to Helmholtz FMM

Back to top

3.3 Stokes FMM

Let $\mathcal{G}^{\text{stok}}(x,y)$ denote the Stokeslet given by

$$\mathcal{G}^{\text{stok}}(x,y) = \frac{1}{8\pi\|x-y\|^3} \begin{bmatrix} (x_1-y_1)^2 + \|x-y\|^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3) \\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 + \|x-y\|^2 & (x_2-y_2)(x_3-y_3) \\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2 + \|x-y\|^2 \end{bmatrix},$$

let $\mathcal{T}^{\text{stok}}(x,y)$ denote the Stresslet whose action on a vector v is given by

$$v \cdot \mathcal{T}^{\text{stok}}(x,y) = \frac{3v\cdot(x-y)}{4\pi\|x-y\|^5} \begin{bmatrix} (x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3) \\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3) \\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2 \end{bmatrix},$$

let $\mathcal{R}^{\text{stok}}(x,y)$ denote the Rotlet whose action on a vector v is given by

$$v \cdot \mathcal{R}^{\text{stok}}(x,y) = \frac{v \cdot (x-y)}{4\pi \|x-y\|^3} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} ,$$

and $\mathcal{D}^{\text{stok}}(x,y)$ denote the symmetric part of Doublet whose action on a vector v is given by

$$v \cdot \mathcal{D}^{\text{stok}}(x,y) = \frac{3v \cdot (x-y)}{4\pi \|x-y\|^5} \begin{bmatrix} (x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3) \\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3) \\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2 \end{bmatrix} - \\ \frac{1}{4\pi \|x-y\|^3} \begin{bmatrix} v_1(x_1-y_1) & v_2(x_1-y_1) & v_3(x_1-y_1) \\ v_2(x_2-y_2) & v_2(x_2-y_2) & v_3(x_2-y_2) \\ v_3(x_3-y_3) & v_3(x_3-y_3) & v_3(x_3-y_3) \end{bmatrix} .$$

The Stokes FMM evaluates the following velocity, its gradient and the associated pressure

$$u(x) = \sum_{m=1}^{N} \mathcal{G}^{\text{stok}}(x,x_j)\sigma_j + \nu_j \cdot \mathcal{T}^{\text{stok}}(x,x_j) \cdot \mu_j + \nu_j^r \cdot \mathcal{R}^{\text{stok}}(x,x_j) \cdot \mu_j^r - \mu_j^r \cdot \mathcal{R}^{\text{stok}}(x,x_j) \cdot \nu_j^r + \\ \nu_j^d \cdot \mathcal{R}^{\text{stok}}(x,x_j) \cdot \mu_j^d - \mu_j^d \cdot \mathcal{R}^{\text{stok}}(x,x_j) \cdot \nu_j^d + \nu_j^d \cdot \mathcal{D}^{\text{stok}}(x,x_j) \cdot \mu_j^d .$$

Here x_j are the source locations, σ_j are the Stokeslet densities, ν_j are the stresslet orientation vectors, μ_j are the stresslet densities, ν_j^r are the rotlet orientation vectors, μ_j^r are the rotlet densities, ν_j^d are the doublet orientation vectors, μ_j^d are the doublet densities, and the locations x at which the velocity and its gradient are evaluated are referred to as the evaluation points.

Unlike the Laplace and Helmholtz FMM, currently we have only the guru interface for the Stokes FMM (for both the single density and the vectorized density cases) with appropriate flags for including or excluding the stokeslet/stresslet term in the interaction, and flags for computing velocity/velocity and pressure/velocity, pressure, and gradients at the evaluation points.

```
subroutine stfmm3d(nd,eps,nsource,source,ifstoklet,stoklet,ifstrslet,strslet,strsvec,
↪ifrotlet,rotlet,rotvec,ifdoublet,doublet,doubvec,ifppreg,pot,pre,grad,ntarg,targ,
↪ifppregtarg,pottarg,pretarg,gradtarg,ier)
```

Input arguments:

- **nd: integer**
  Number of densities
- **eps: double precision**
  Precision requested
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **ifstoklet: integer**
  Flag for including Stokeslet (σ_j) term in interaction kernel Stokeslet term will be included if ifstoklet = 1
- **stoklet: double precision(nd,3,nsource)**
  Stokeslet strengths, σ_j
- **ifstrslet: integer**
  Flag for including Stresslet (μ_j, ν_j) term in interaction kernel Stresslet term will be included if ifstrslet = 1
- **strslet: double precision(nd,3,nsource)**
  Stresslet strengths, μ_j

- **strsvec: double precision(nd,3,nsource)**
  Stresslet orientation vectors, ν_j
- **ifrotlet: integer**
  Flag for including Rotlet (μ_j^r, ν_j^r) term in interaction kernel Rotlet term will be included if ifrotlet = 1
- **rotlet: double precision(nd,3,nsource)**
  Rotlet strengths, μ_j^r
- **rotvec: double precision(nd,3,nsource)**
  Rotlet orientation vectors, ν_j^r
- **ifdoublet: integer**
  Flag for including Doublet (μ_j^d, ν_j^d) term in interaction kernel Doublet term will be included if ifdoublet = 1
- **doublet: double precision(nd,3,nsource)**
  Doublet strengths, μ_j^d
- **doubvec: double precision(nd,3,nsource)**
  Doublet orientation vectors, ν_j^d
- **ifppreg: integer**

  Flag for computing velocity, pressure and/or gradients at source locations

  ifppreg = 1, compute velocity

  ifppreg = 2, compute velocity and pressure

  ifppreg = 3, compute veloicty, pressure and gradient
- **ntarg: integer**
  Number of targets
- **targets: double precision (3,ntarg)**
  Target locations x
- **ifppregtarg: integer**

  Flag for computing velocity, pressure and/or gradients at target locations

  ifppregtarg = 1, compute velocity

  ifppregtarg = 2, compute velocity and pressure

  ifppregtarg = 3, compute veloicty, pressure and gradient

Output arguments:

- **pot: double precision (nd,3,nsource)**
  Velocity at source locations if requested
- **pre: double precision (nd,nsource)**
  Pressure at source locations if requested
- **grad: double precision (nd,3,3,nsource)**
  Gradient at source locations if requested
- **pottarg: double precision (nd,3,ntarg)**
  Velocity at target locations if requested
- **pretarg: double precision (nd,ntarg)**
  Pressure at target locations if requested
- **gradtarg: double precision (nd,3,3,ntarg)**
  Gradient at target locations if requested

- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Example drivers:

- `examples/stfmm3d_example.f`. The corresponding makefile is `examples/stfmm3d_example.make`

Back to top

3.4 Maxwell FMM

The Maxwell FMM evaluates the following field, its curl, and its divergence

$$E(x) = \sum_{j=1}^{N} \nabla \times \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} M_j + \frac{e^{ik\|x-x_j\|}}{\|x-x_j\|} J_j + \nabla \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \rho_j \,.$$

Here x_j are the source locations, M_j are the magnetic current densities, J_j are the electric current densities, ρ_j are the electric charge densities, and the collection of x at which the field, its curl and its divergence are evalauated are referred to as the evaluation points.

Unlike the Laplace and Helmholtz FMM, currently we have only the guru interface for the Maxwell FMM (for both the single density and the vectorized density cases) with appropriate flags for including or excluding the magnetic current/electric current/electric charge term in the interaction, and flags for computing field/curl/divergence at the evaluation points.

```
subroutine emfmm3d(nd,eps,zk,ns,source,ifh_current,h_current,ife_current,e_current,ife_
↪charge,e_charge,nt,targets,ifE,E,ifcurlE,curlE,ifdivE,divE,ier)
```

Input arguments:

- **nd: integer**
  Number of densities
- **eps: double precision**
  Precision requested
- **zk: double complex**
  Wave number k
- **ns: integer**
  Number of sources
- **source: double precision(3,ns)**
  Source locations, x_j
- **ifh_current: integer**
  Flag for including magnetic current (M_j) term in interaction kernel. Magnetic current term will be included if ifh_current = 1
- **h_current: double complex(nd,3,ns)**
  Magnetic currents, M_j
- **ife_current: integer**
  Flag for including electric current (J_j) term in interaction kernel. Electric current term will be included if ife_current = 1
- **e_current: double complex(nd,3,ns)**
  Electric currents, J_j

- **ife_charge: integer**
  Flag for including electric charge (ρ_j) term in interaction kernel. Electric charge term will be included if ife_charge = 1
- **e_charge: double complex(nd,ns)**
  Electric charges, ρ_j
- **nt: integer**
  Number of targets
- **targets: double precision (3,nt)**
  Target locations x
- **ifE: integer**
  Flag for computing field. The field E will be returned if ifE = 1
- **ifcurlE: integer**
  Flag for computing curl of field. $\nabla \times E$ will be returned if ifcurlE = 1
- **ifdivE: integer**
  Flag for computing divergence of field. $\nabla \cdot E$ will be returned if ifdivE = 1

Output arguments:

- **E: double complex (nd,3,nt)**
  Field at the evaluation points if requested
- **curlE: double complex (nd,3,nt)**
  Curl of field at the evaluation points if requested
- **divE: double complex (nd,nt)**
  Divergence of field at the evaluation points if requested
- **ier: integer**
  Error flag; ier=0 implies successful execution, and ier=4/8 implies insufficient memory

Example drivers:

- `examples/emfmm3d_example.f`. The corresponding makefile is `examples/emfmm3d_example.make`

Back to top

3.4.1 List of interfaces

Back to top

3.5 C interfaces

All of the above fortran routines can be called from c by including the header `utils.h` and `lfmm3d_c.h` for Laplace FMMs or `hfmm3d_c.h` for Helmholtz FMMs.

For example, the subroutine to evaluate the potential and gradient, at a collection of targets t_i due to a collection of Helmholtz charges is:

```
hfmm3d_t_c_g_
```

In general, to call a fortran subroutine from `c` use:

```
"<fortran subroutine name>"_("<calling sequence>")
```

> **Note**
>
> All the variables in the calling sequence must be passed as pointers from `c`.

> **Note**
>
> For the vectorized subroutines, the charge strengths, dipole strengths, potentials, and gradients are interleaved as opposed to provided in a sequential manner. For example for three sets of charge strengths, they should be stored as $c_{1,1}, c_{2,1}, c_{3,1}, c_{1,2}, c_{2,2}, c_{3,2} \ldots c_{1,N}, c_{2,N}, c_{3,N}$.

Example drivers:

- Laplace:
  - `c/lfmm3d_example.c`. The corresponding makefile is `c/lfmm3d_example.make`
  - `c/lfmm3d_vec_example.c`. The corresponding makefile is `c/lfmm3d_vec_example.make`
- Helmholtz:
  - `c/hfmm3d_example.c`. The corresponding makefile is `c/hfmm3d_example.make`
  - `c/hfmm3d_vec_example.c`. The corresponding makefile is `c/hfmm3d_vec_example.make`

The Maxwell and Stokes interfaces are currently unavailable in C, and will be made available soon.

Back to top

CHAPTER

FOUR

MATLAB

The MATLAB interface has four callable subroutines:

- Laplace wrappers: Fast multipole implementation (lfmm3d) and direct evaluation (l3ddir) for Laplace N-body interactions
- Helmholtz wrappers: Fast multipole implementation (hfmm3d) and direct evaluation (h3ddir) for Helmholtz N-body interactions
- Stokes wrappers: Fast multipole implementation (stfmm3d) and direct evaluation (st3ddir) for Stokes N-body interactions
- Maxwell wrappers: Fast multipole implementation (emfmm3d) and direct evaluation (em3ddir) for Maxwell N-body interactions

4.1 Laplace wrappers

This subroutine computes the N-body Laplace interactions and its gradients in three dimensions where the interaction kernel is given by $1/r$

$$u(x) = \sum_{j=1}^{N} \frac{c_j}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
function [U] = lfmm3d(eps,srcinfo,pg,targ,pgt)
```

Wrapper for fast multipole implementation for Laplace N-body interactions.

Args:

- **eps: double**
  precision requested
- **srcinfo: structure**
  structure containing sourceinfo
  - **srcinfo.sources: double(3,n)**
    source locations, x_j
  - **srcinfo.nd: integer**
    number of charge/dipole vectors (optional, default - nd = 1)
  - **srcinfo.charges: double(nd,n)**
    charge densities, c_j (optional, default - term corresponding to charges dropped)

  - **srcinfo.dipoles: double(nd,3,n)**
    dipole orientation vectors, v_j (optional default - term corresponding to dipoles dropped)

- **pg: integer**

  source eval flag
  potential at sources evaluated if pg = 1
  potenial and gradient at sources evaluated if pg=2

- **targ: double(3,nt)**
  target locations, t_i (optional)

- **pgt: integer**

  target eval flag (optional)
  potential at targets evaluated if pgt = 1
  potenial and gradient at targets evaluated if pgt=2

Returns:

- U.pot: potential at source locations, if requested, $u(x_j)$
- U.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- U.pottarg: potential at target locations, if requested, $u(t_i)$
- U.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$

Wrapper for direct evaluation of Laplace N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
function [U] = l3ddir(srcinfo,targ,pgt)
```

Example:

- see `lfmm3d_example.m`

Back to top

4.2 Helmholtz wrappers

This subroutine computes the N-body Helmholtz interactions and its gradients in three dimensions where the interaction kernel is given by e^{ikr}/r

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
function [U] = hfmm3d(eps,zk,srcinfo,pg,targ,pgt)
```

Wrapper for fast multipole implementation for Helmholtz N-body interactions.

Args:

- **eps: double**
  precision requested

- **zk: complex**
  Helmholtz parameter, k
- **srcinfo: structure**
  structure containing sourceinfo
  - **srcinfo.sources: double(3,n)**
    source locations, x_j
  - **srcinfo.nd: integer**
    number of charge/dipole vectors (optional, default - nd = 1)
  - **srcinfo.charges: complex(nd,n)**
    charge densities, c_j (optional, default - term corresponding to charges dropped)
  - **srcinfo.dipoles: complex(nd,3,n)**
    dipole orientation vectors, v_j (optional default - term corresponding to dipoles dropped)
- **pg: integer**

  source eval flag

  potential at sources evaluated if pg = 1

  potenial and gradient at sources evaluated if pg=2
- **targ: double(3,nt)**
  target locations, t_i (optional)
- **pgt: integer**

  target eval flag (optional)

  potential at targets evaluated if pgt = 1

  potenial and gradient at targets evaluated if pgt=2

Returns:

- U.pot: potential at source locations, if requested, $u(x_j)$
- U.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- U.pottarg: potential at target locations, if requested, $u(t_i)$
- U.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$

Wrapper for direct evaluation of Helmholtz N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
function [U] = h3ddir(zk,srcinfo,targ,pgt)
```

Example:

- see `hfmm3d_example.m`

Back to top

4.3 Stokes wrappers

Let $\mathcal{G}^{\text{stok}}(x,y)$ denote the Stokeslet given by

$$\mathcal{G}^{\text{stok}}(x,y)=\frac{1}{8\pi\|x-y\|^3}\begin{bmatrix}(x_1-y_1)^2+\|x-y\|^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2+\|x-y\|^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2+\|x-y\|^2\end{bmatrix},$$

let $\mathcal{T}^{\text{stok}}(x,y)$ denote the Stresslet whose action on a vector v is given by

$$v\cdot\mathcal{T}^{\text{stok}}(x,y)=\frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix},$$

let $\mathcal{R}^{\text{stok}}(x,y)$ denote the Rotlet whose action on a vector v is given by

$$v\cdot\mathcal{R}^{\text{stok}}(x,y)=\frac{v\cdot(x-y)}{4\pi\|x-y\|^3}\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix},$$

and $\mathcal{D}^{\text{stok}}(x,y)$ denote the symmetric part of Doublet whose action on a vector v is given by

$$v\cdot\mathcal{D}^{\text{stok}}(x,y)=\frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix}-\frac{1}{4\pi\|x-y\|^3}\begin{bmatrix}v_1(x_1-y_1) & v_2(x_1-y_1) & v_3(x_1-y_1)\\ v_2(x_2-y_2) & v_2(x_2-y_2) & v_3(x_2-y_2)\\ v_3(x_3-y_3) & v_3(x_3-y_3) & v_3(x_3-y_3)\end{bmatrix}.$$

This subroutine computes the N-body Stokes interactions, its gradients and the corresponding pressure in three dimensions given by

$$u(x)=\sum_{m=1}^{N}\mathcal{G}^{\text{stok}}(x,x_j)\sigma_j+\nu_j\cdot\mathcal{T}^{\text{stok}}(x,x_j)\cdot\mu_j+\nu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^r-\mu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^r+\nu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^d-\mu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^d+\nu_j^d\cdot\mathcal{D}^{\text{stok}}(x,x_j)\cdot\mu_j^d.$$

where σ_j are the Stokeslet densities, ν_j are the stresslet orientation vectors, μ_j are the stresslet densities, ν_j^r are the rotlet orientation vectors, μ_j^r are the rotlet densities, ν_j^d are the doublet orientation vectors, μ_j^d are the doublet densities, and x_j are the source locations. When $x=x_j$, the term corresponding to x_j is dropped from the sum.

```
function [U] = stfmm3d(eps,srcinfo,ifppreg,targ,ifppregtarg)
```

Wrapper for fast multipole implementation for Stokes N-body interactions.

Args:

- **eps: double**
  precision requested
- **srcinfo: structure**
  structure containing sourceinfo
  - **srcinfo.sources: double(3,n)**
    source locations, x_j
  - **srcinfo.nd: integer**
    number of charge/dipole vectors (optional, default - nd = 1)

- **srcinfo.stoklet: double(nd,3,n)**
  Stokeslet densities, σ_j (optional, default - term corresponding to Stokeslet dropped)
- **srcinfo.strslet: double(nd,3,n)**
  Stresslet densities, μ_j (optional default - term corresponding to stresslet dropped)
- **srcinfo.strsvec: double(nd,3,n)**
  Stresslet orientation vectors, ν_j (optional default - term corresponding to stresslet dropped)
- **srcinfo.rotlet: double(nd,3,n)**
  Rotlet densities, μ_j^r (optional default - term corresponding to rotlet dropped)
- **srcinfo.rotvec: double(nd,3,n)**
  Rotlet orientation vectors, ν_j^r (optional default - term corresponding to rotlet dropped)
- **srcinfo.doublet: double(nd,3,n)**
  Doublet densities, μ_j^d (optional default - term corresponding to doublet dropped)
- **srcinfo.doubvec: double(nd,3,n)**
  Doublet orientation vectors, ν_j^d (optional default - term corresponding to doublet dropped)

- **ifppreg: integer**

  source eval flag
  potential at sources evaluated if ifppreg = 1
  potential and pressure at sources evaluated if ifppreg=2
  potential, pressure and gradient at sources evaluated if ifppreg=3

- **targ: double(3,nt)**
  target locations, t_i (optional)

- **ifppregtarg: integer**

  target eval flag (optional)
  potential at targets evaluated if ifppregtarg = 1
  potential and pressure at targets evaluated if ifppregtarg = 2
  potential, pressure and gradient at targets evaluated if ifppregtarg = 3

Returns:

- U.pot: velocity at source locations if requested
- U.pre: pressure at source locations if requested
- U.grad: gradient of velocity at source locations if requested
- U.pottarg: velocity at target locations if requested
- U.pretarg: pressure at target locations if requested
- U.gradtarg: gradient of velocity at target locations if requested

---

Wrapper for direct evaluation of Stokes N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
function [U] = st3ddir(srcinfo,targ,ifppregtarg)
```

Example:

- see `stfmm3d_example.m`

Back to top

4.4 Maxwell wrappers

This subroutine computes the N-body Maxwell interactions, its curl and its divergence in three dimensions given by

$$E(x) = \sum_{j=1}^{N} \nabla \times \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} M_j + \frac{e^{ik\|x-x_j\|}}{\|x-x_j\|} J_j + \nabla \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \rho_j$$

where M_j are the magnetic current densities, J_j are the electric current densities, ρ_j are the electric charge densities, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
function [U] = emfmm3d(eps,zk,srcinfo,targ,ifE,ifcurlE,ifdivE)
```

Wrapper for fast multipole implementation for Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

Args:

- **eps: double**
  precision requested
- **zk: complex**
  Wavenumber, k
- **srcinfo: structure**
  structure containing sourceinfo
  - **srcinfo.sources: double(3,n)**
    source locations, x_j
  - **srcinfo.nd: integer**
    number of charge/dipole vectors (optional, default - nd = 1)
  - **srcinfo.h_current: complex(nd,3,n)**
    Magnetic current densities, M_j (optional, default - term corresponding to magnetic current dropped)
  - **srcinfo.e_current: complex(nd,3,n)**
    Electric current densities, J_j (optional, default - term corresponding to electric current dropped)
  - **srcinfo.e_charge: complex(nd,n)**
    Electric charge densities, ρ_j (optional, default - term corresponding to electric charge dropped)
- **targ: double(3,nt)**
  target locations, t_i
- **ifE: integer**
  E is returned at the target locations if ifE = 1
- **ifcurlE: integer**
  curl E is returned at the target locations if ifcurlE = 1
- **ifdivE: integer**
  div E is returned at the target locations if ifdivE = 1

Returns:

- U.E: E field defined above at target locations if requested $(E(t_j))$
- U.curlE: curl of E field at target locations if requested $(\nabla \times E(t_j))$

- U.divE: divergence of E at target locations if requested $(\nabla \cdot E(t_j))$

Wrapper for direct evaluation of Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

```
function [U] = em3ddir(zk,srcinfo,targ,ifE,ifcurlE,ifdivE)
```

Example:

- see `emfmm3d_example.m`

Back to top

CHAPTER

FIVE

PYTHON

The Python interface has four callable subroutines:

- Laplace wrappers: Fast multipole implementation (lfmm3d) and direct evaluation (l3ddir) for Laplace N-body interactions
- Helmholtz wrappers: Fast multipole implementation (hfmm3d) and direct evaluation (h3ddir) for Helmholtz N-body interactions
- Stokes wrappers: Fast multipole implementation (stfmm3d) and direct evaluation (st3ddir) for Stokes N-body interactions
- Maxwell wrappers: Fast multipole implementation (emfmm3d) and direct evaluation (em3ddir) for Maxwell N-body interactions

5.1 Laplace wrappers

This subroutine computes the N-body Laplace interactions and its gradients in three dimensions where the interaction kernel is given by $\frac{1}{l}4\pi r$

$$u(x) = \sum_{j=1}^{N} \frac{c_j}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
def lfmm3d(*,eps,sources,charges=None,dipvec=None,targets=None,pg=0,pgt=0,nd=1)
```

Wrapper for fast multipole implementation for Laplace N-body interactions.

Args:

- **eps: double**
  precision requested
- **sources: double(3,n)**
  source locations, x_j
- **charges: double(n,) or double(nd,n)**
  charge densities, c_j
- **dipvec: double(3,n) or double(nd,3,n)**
  dipole orientation vectors, v_j
- **nd: integer**
  number of charge/dipole vectors

- **pg: integer**

  source eval flag

  potential at sources evaluated if pg = 1

  potenial and gradient at sources evaluated if pg=2

- **targets: double(3,nt)**
  target locations (t_i) (optional)

- **pgt: integer**

  target eval flag (optional)

  potential at targets evaluated if pgt = 1

  potenial and gradient at targets evaluated if pgt=2

Returns: The subroutine returns an object out of type Output with the following variables

- out.pot: potential at source locations, if requested, $u(x_j)$
- out.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- out.pottarg: potential at target locations, if requested, $u(t_i)$
- out.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$

Wrapper for direct evaluation of Laplace N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
def l3ddir(*,sources,charges=None,dipvec=None,targets=None,pgt=0,nd=1)
```

Example:

- see `lfmm3d_example.py`

Back to top

5.2 Helmholtz wrappers

This subroutine computes the N-body Helmholtz interactions and its gradients in three dimensions where the interaction kernel is given by $\frac{e^{ikr}}{4\pi r}$

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
def hfmm3d(*,eps,zk,sources,charges=None,dipvec=None,targets=None,pg=0,pgt=0,nd=1)
```

Wrapper for fast multipole implementation for Helmholtz N-body interactions.

Args:

- **eps: double**
  precision requested

- **zk: complex**
  Helmholtz parameter, k
- **sources: double(3,n)**
  source locations, x_j
- **charges: complex(n,) or complex(nd,n)**
  charge densities, c_j
- **dipvec: complex(3,n) or complex(nd,3,n)**
  dipole orientation vectors, v_j
- **nd: integer**
  number of charge/dipole vectors
- **pg: integer**

  source eval flag

  potential at sources evaluated if pg = 1

  potenial and gradient at sources evaluated if pg=2
- **targets: double(3,nt)**
  target locations, t_i (optional)
- **pgt: integer**

  target eval flag (optional)

  potential at targets evaluated if pgt = 1

  potenial and gradient at targets evaluated if pgt=2

Returns: The subroutine returns an object out of type Output with the following variables

- out.pot: potential at source locations, if requested, $u(x_j)$
- out.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- out.pottarg: potential at target locations, if requested, $u(t_i)$
- out.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$

Wrapper for direct evaluation of Helmholtz N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
def h3ddir(*,zk,sources,charges=None,dipvec=None,targets=None,pgt=0,nd=1)
```

Example:

- see `hfmm3d_example.py`

Back to top

5.3 Stokes wrappers

Let $\mathcal{G}^{\text{stok}}(x,y)$ denote the Stokeslet given by

$$\mathcal{G}^{\text{stok}}(x,y) = \frac{1}{8\pi\|x-y\|^3}\begin{bmatrix} (x_1-y_1)^2+\|x-y\|^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3) \\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2+\|x-y\|^2 & (x_2-y_2)(x_3-y_3) \\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2+\|x-y\|^2 \end{bmatrix},$$

let $\mathcal{T}^{\text{stok}}(x,y)$ denote the Stresslet whose action on a vector v is given by

$$v \cdot \mathcal{T}^{\text{stok}}(x,y) = \frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\(x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\(x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix},$$

let $\mathcal{R}^{\text{stok}}(x,y)$ denote the Rotlet whose action on a vector v is given by

$$v \cdot \mathcal{R}^{\text{stok}}(x,y) = \frac{v\cdot(x-y)}{4\pi\|x-y\|^3}\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix},$$

and $\mathcal{D}^{\text{stok}}(x,y)$ denote the symmetric part of Doublet whose action on a vector v is given by

$$v \cdot \mathcal{D}^{\text{stok}}(x,y) = \frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\(x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\(x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix} - \frac{1}{4\pi\|x-y\|^3}\begin{bmatrix}v_1(x_1-y_1) & v_2(x_1-y_1) & v_3(x_1-y_1)\\v_2(x_2-y_2) & v_2(x_2-y_2) & v_3(x_2-y_2)\\v_3(x_3-y_3) & v_3(x_3-y_3) & v_3(x_3-y_3)\end{bmatrix}.$$

This subroutine computes the N-body Stokes interactions, its gradients and the corresponding pressure in three dimensions given by

$$u(x) = \sum_{m=1}^{N} \mathcal{G}^{\text{stok}}(x,x_j)\sigma_j + \nu_j\cdot\mathcal{T}^{\text{stok}}(x,x_j)\cdot\mu_j + \nu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^r - \mu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^r + \nu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^d - \mu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^d + \nu_j^d\cdot\mathcal{D}^{\text{stok}}(x,x_j)\cdot\mu_j^d\,.$$

where σ_j are the Stokeslet densities, ν_j are the stresslet orientation vectors, μ_j are the stresslet densities, ν_j^r are the rotlet orientation vectors, μ_j^r are the rotlet densities, ν_j^d are the doublet orientation vectors, μ_j^d are the doublet densities, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
def stfmm3d(*,eps,sources,stoklet=None,strslet=None,strsvec=None,rotlet=None,rotvec=None,
↪doublet=None,doubvec=None,targets=None,ifppreg=0,ifppregtarg=0,nd=1)
```

Wrapper for fast multipole implementation for Stokes N-body interactions.

Args:

- **eps: double**
  precision requested
- **sources: float(3,n)**
  source locations
- **stoklet: float(nd,3,n) or float(3,n)**
  Stokeslet charge strengths (σ_j above)
- **strslet: float(nd,3,n) or float(3,n)**
  stresslet strengths (μ_j above)
- **strsvec: float(nd,3,n) or float(3,n)**
  stresslet orientations (ν_j above)
- **rotlet: float(nd,3,n) or float(3,n)**
  rotlet strengths (μ_j^r above)
- **rotvec: float(nd,3,n) or float(3,n)**
  rotlet orientations (ν_j^r above)

- **doublet: float(nd,3,n) or float(3,n)**
  doublet strengths (μ_j^d above)
- **doubvec: float(nd,3,n) or float(3,n)**
  doublet orientations (ν_j^d above)
- **targets: float(3,nt)**
  target locations (x)
- **ifppreg: integer**
  flag for evaluating potential, gradient, and pressure at sources
  potential at sources evaluated if ifppreg = 1
  potential and pressure at sources evaluated if ifppreg=2
  potential, pressure and gradient at sources evaluated if ifppreg=3
- **ifppregtarg: integer**
  flag for evaluating potential, gradient, and pressure at targets
  potential at targets evaluated if ifppregtarg = 1
  potential and pressure at targets evaluated if ifppregtarg = 2
  potential, pressure and gradient at targets evaluated if ifppregtarg = 3

Returns:

- out.pot: velocity at source locations if requested
- out.pre: pressure at source locations if requested
- out.grad: gradient of velocity at source locations if requested
- out.pottarg: velocity at target locations if requested
- out.pretarg: pressure at target locations if requested
- out.gradtarg: gradient of velocity at target locations if requested

---

Wrapper for direct evaluation of Stokes N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
def st3ddir(*,eps,sources,stoklet=None,strslet=None,strsvec=None,rotlet=None,rotvec=None,
↪doublet=None,doubvec=None,targets=None,ifppreg=0,ifppregtarg=0,nd=1,thresh=1e-16):
```

Example:

- see `stfmm3d_example.py`

Back to top

5.4 Maxwell wrappers

This subroutine computes the N-body Maxwell interactions, its curl and its divergence in three dimensions given by

$$E(x) = \sum_{j=1}^{N} \nabla \times \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} M_j + \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} J_j + \nabla \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \rho_j$$

where M_j are the magnetic current densities, J_j are the electric current densities, ρ_j are the electric charge densities, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
def emfmm3d(*,eps,zk,sources,h_current=None,e_current=None,e_charge=None,targets=None,
↪ifE=0,ifcurlE=0,ifdivE=0,nd=1):
```

Wrapper for fast multipole implementation for Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

Args:

- **eps: double**
  precision requested
- **zk: complex**
  Wavenumber, k
- **sources: float(3,n)**
  source locations
- **h_current: complex(3,n) or complex(nd,3,n)**
  Magnetic currents, M_j
- **e_current: complex(3,n) or complex(nd,3,n)**
  Electric currents, J_j
- **e_charge: complex(n,) or complex(nd,n)**
  Electric charges, ρ_j
- **targets: float(3,nt)**
  target locations, t_i
- **ifE: integer**
  E is returned at the target locations if ifE = 1
- **ifcurlE: integer**
  curl E is returned at the target locations if ifcurlE = 1
- **ifdivE: integer**
  div E is returned at the target locations if ifdivE = 1

Returns:

- out.E: E field defined above at target locations if requested $(E(t_j))$
- out.curlE: curl of E field at target locations if requested $(\nabla \times E(t_j))$
- out.divE: divergence of E at target locations if requested $(\nabla \cdot E(t_j))$

Wrapper for direct evaluation of Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

```
def em3ddir(*,eps,zk,sources,h_current=None,e_current=None,e_charge=None,targets=None,
↪ifE=0,ifcurlE=0,ifdivE=0,nd=1,thresh=1e-16):
```

Example:

- see `emfmm3d_example.py`

Back to top

CHAPTER

SIX

JULIA

The julia interface has high-level subroutines for four interaction kernels:

- Laplace wrappers: Fast multipole implementation (lfmm3d) and direct evaluation (l3ddir) for Laplace N-body interactions
- Helmholtz wrappers: Fast multipole implementation (hfmm3d) and direct evaluation (h3ddir) for Helmholtz N-body interactions
- Stokes wrappers: Fast multipole implementation (stfmm3d) and direct evaluation (st3ddir) for Stokes N-body interactions
- Maxwell wrappers: Fast multipole implementation (emfmm3d) and direct evaluation (em3ddir) for Maxwell N-body interactions

6.1 Laplace wrappers

This subroutine computes the N-body Laplace interactions and its gradients in three dimensions where the interaction kernel is given by $1/(4\pi r)$

$$u(x) = \sum_{j=1}^{N} \frac{c_j}{4\pi\|x - x_j\|} - v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
vals = lfmm3d(eps,sources;charges=nothing,dipvecs=nothing,
              targets=nothing,pg=0,pgt=0,nd=1)
```

Wrapper for fast multipole implementation for Laplace N-body interactions.

Args:

- **eps: double**
  precision requested
- **sources: double(3,n)**
  source locations, x_j
- **charges: double(n,) or double(nd,n)**
  charge densities, c_j
- **dipvec: double(3,n) or double(nd,3,n)**
  dipole orientation vectors, v_j
- **nd: integer**
  number of charge/dipole vectors

- **pg: integer**

  source eval flag

  potential at sources evaluated if pg = 1

  potenial and gradient at sources evaluated if pg=2

- **targets: double(3,nt)**

  target locations (t_i) (optional)

- **pgt: integer**

  target eval flag (optional)

  potential at targets evaluated if pgt = 1

  potenial and gradient at targets evaluated if pgt=2

Returns: The subroutine returns an object val of type FMMVals with the following variables

- vals.pot: potential at source locations, if requested, $u(x_j)$
- vals.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- vals.pottarg: potential at target locations, if requested, $u(t_i)$
- vals.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$
- vals.ier: error flag as returned by FMM3D library. A value of 0 indicates a successful call. Non-zero values may indicate insufficient memory available. See the documentation for the FMM3D library.

Wrapper for direct evaluation of Laplace N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
vals = l3ddir(sources,targets;charges=nothing,
              dipvecs=nothing,pgt=0,nd=1,
              thresh=1e-16)
```

Example:

- see `lfmmexample.jl`

Back to top

6.2 Helmholtz wrappers

This subroutine computes the N-body Helmholtz interactions and its gradients in three dimensions where the interaction kernel is given by e^{ikr}/r

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)$$

where c_j are the charge densities v_j are the dipole orientation vectors, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
vals = hfmm3d(eps,zk,sources;charges=nothing,dipvecs=nothing,
               targets=nothing,pg=0,pgt=0,nd=1)
```

Wrapper for fast multipole implementation for Helmholtz N-body interactions.

Args:

- **eps: double**
  precision requested
- **zk: complex**
  Helmholtz parameter, k
- **sources: double(3,n)**
  source locations, x_j
- **charges: complex(n,) or complex(nd,n)**
  charge densities, c_j
- **dipvec: complex(3,n) or complex(nd,3,n)**
  dipole orientation vectors, v_j
- **nd: integer**
  number of charge/dipole vectors
- **pg: integer**

  source eval flag

  potential at sources evaluated if pg = 1

  potenial and gradient at sources evaluated if pg=2
- **targets: double(3,nt)**
  target locations, t_i (optional)
- **pgt: integer**

  target eval flag (optional)

  potential at targets evaluated if pgt = 1

  potenial and gradient at targets evaluated if pgt=2

Returns: The subroutine returns an object vals of type FMMVals with the following variables

- vals.pot: potential at source locations, if requested, $u(x_j)$
- vals.grad: gradient at source locations, if requested, $\nabla u(x_j)$
- vals.pottarg: potential at target locations, if requested, $u(t_i)$
- vals.gradtarg: gradient at target locations, if requested, $\nabla u(t_i)$

Wrapper for direct evaluation of Helmholtz N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
vals = h3ddir(zk,sources,targets;charges=nothing,
              dipvecs=nothing,pgt=0,nd=1,
              thresh=1e-16)
```

Example:

- see `hfmmexample.jl`

Back to top

6.3 Stokes wrappers

Let $\mathcal{G}^{\text{stok}}(x,y)$ denote the Stokeslet given by

$$\mathcal{G}^{\text{stok}}(x,y) = \frac{1}{8\pi\|x-y\|^3}\begin{bmatrix}(x_1-y_1)^2+\|x-y\|^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2+\|x-y\|^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2+\|x-y\|^2\end{bmatrix},$$

let $\mathcal{T}^{\text{stok}}(x,y)$ denote the Stresslet whose action on a vector v is given by

$$v\cdot\mathcal{T}^{\text{stok}}(x,y) = \frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix},$$

let $\mathcal{R}^{\text{stok}}(x,y)$ denote the Rotlet whose action on a vector v is given by

$$v\cdot\mathcal{R}^{\text{stok}}(x,y) = \frac{v\cdot(x-y)}{4\pi\|x-y\|^3}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix},$$

and $\mathcal{D}^{\text{stok}}(x,y)$ denote the symmetric part of Doublet whose action on a vector v is given by

$$v\cdot\mathcal{D}^{\text{stok}}(x,y) = \frac{3v\cdot(x-y)}{4\pi\|x-y\|^5}\begin{bmatrix}(x_1-y_1)^2 & (x_1-y_1)(x_2-y_2) & (x_1-y_1)(x_3-y_3)\\ (x_2-y_2)(x_1-y_1) & (x_2-y_2)^2 & (x_2-y_2)(x_3-y_3)\\ (x_3-y_3)(x_1-y_1) & (x_3-y_3)(x_2-y_2) & (x_3-y_3)^2\end{bmatrix} - \frac{1}{4\pi\|x-y\|^3}\begin{bmatrix}v_1(x_1-y_1) & v_2(x_1-y_1) & v_3(x_1-y_1)\\ v_2(x_2-y_2) & v_2(x_2-y_2) & v_3(x_2-y_2)\\ v_3(x_3-y_3) & v_3(x_3-y_3) & v_3(x_3-y_3)\end{bmatrix}.$$

This subroutine computes the N-body Stokes interactions, its gradients and the corresponding pressure in three dimensions given by

$$u(x) = \sum_{m=1}^{N}\mathcal{G}^{\text{stok}}(x,x_j)\sigma_j + \nu_j\cdot\mathcal{T}^{\text{stok}}(x,x_j)\cdot\mu_j + \nu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^r - \mu_j^r\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^r + \nu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\mu_j^d - \mu_j^d\cdot\mathcal{R}^{\text{stok}}(x,x_j)\cdot\nu_j^d + \nu_j^d\cdot\mathcal{D}^{\text{stok}}(x,x_j)\cdot\mu_j^d.$$

where σ_j are the Stokeslet densities, ν_j are the stresslet orientation vectors, μ_j are the stresslet densities, ν_j^r are the rotlet orientation vectors, μ_j^r are the rotlet densities, ν_j^d are the doublet orientation vectors, μ_j^d are the doublet densities, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
vals = stfmm3d(eps,sources;stoklet=nothing,strslet=nothing,
               strsvec=nothing,rotlet=nothing,rotvec=nothing,
               doublet=nothing,doubvec=nothing,targets=nothing,ppreg=0,
               ppregt=0,nd=1)
```

Wrapper for fast multipole implementation for Stokes N-body interactions.

Args:

- **eps: double**
  precision requested
- **sources: float(3,n)**
  source locations
- **stoklet: float(nd,3,n) or float(3,n)**
  Stokeslet charge strengths (σ_j above)

- **strslet: float(nd,3,n) or float(3,n)**
  stresslet strengths (mu_j above)
- **strsvec: float(nd,3,n) or float(3,n)**
  stresslet orientations (nu_j above)
- **rotlet: float(nd,3,n) or float(3,n)**
  rotlet strengths (mu_j^r above)
- **rotvec: float(nd,3,n) or float(3,n)**
  rotlet orientations (nu_j^r above)
- **doublet: float(nd,3,n) or float(3,n)**
  doublet strengths (mu_j^d above)
- **doubvec: float(nd,3,n) or float(3,n)**
  doublet orientations (nu_j^d above)
- **targets: float(3,nt)**
  target locations (x)
- **ifppreg: integer**

  flag for evaluating potential, gradient, and pressure at sources
  potential at sources evaluated if ifppreg = 1
  potential and pressure at sources evaluated if ifppreg=2
  potential, pressure and gradient at sources evaluated if ifppreg=3
- **ifppregtarg: integer**

  flag for evaluating potential, gradient, and pressure at targets
  potential at targets evaluated if ifppregtarg = 1
  potential and pressure at targets evaluated if ifppregtarg = 2
  potential, pressure and gradient at targets evaluated if ifppregtarg = 3

Returns:

- vals.pot: velocity at source locations if requested
- vals.pre: pressure at source locations if requested
- vals.grad: gradient of velocity at source locations if requested
- vals.pottarg: velocity at target locations if requested
- vals.pretarg: pressure at target locations if requested
- vals.gradtarg: gradient of velocity at target locations if requested

---

Wrapper for direct evaluation of Stokes N-body interactions. Note that this wrapper only returns potentials and gradients at the target locations.

```
vals = st3ddir(sources,targets;stoklet=nothing,strslet=nothing,
               strsvec=nothing,rotlet=nothing,rotvec=nothing,
               doublet=nothing,doubvec=nothing,ppregt=0,nd=1,thresh=1e-16)
```

Back to top

6.4 Maxwell wrappers

This subroutine computes the N-body Maxwell interactions, its curl and its divergence in three dimensions given by

$$E(x) = \sum_{j=1}^{N} \nabla \times \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} M_j + \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} J_j + \nabla \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \rho_j$$

where M_j are the magnetic current densities, J_j are the electric current densities, ρ_j are the electric charge densities, and x_j are the source locations. When $x = x_j$, the term corresponding to x_j is dropped from the sum.

```
vals = emfmm3d(eps,zk,sources;h_current=nothing,e_current=nothing,e_charge=nothing,
            ifE=false,ifdivE=false,ifcurlE=false,
            ifEtarg=false,ifdivEtarg=false,ifcurlEtarg=false,
            nd=1,targets=nothing)
```

Wrapper for fast multipole implementation for Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

Args:

- **eps: double**
  precision requested
- **zk: complex**
  Wavenumber, k
- **sources: float(3,n)**
  source locations
- **h_current: complex(3,n) or complex(nd,3,n)**
  Magnetic currents, M_j
- **e_current: complex(3,n) or complex(nd,3,n)**
  Electric currents, J_j
- **e_charge: complex(n,) or complex(nd,n)**
  Electric charges, ρ_j
- **targets: float(3,nt)**
  target locations, t_i
- **ifE: boolean**
  E is returned at the source locations if ifE = true
- **ifcurlE: boolean**
  curl E is returned at the source locations if ifcurlE = true
- **ifdivE: boolean**
  div E is returned at the source locations if ifdivE = true
- **ifEtarg: boolean**
  E is returned at the target locations if ifE = true
- **ifcurlEtarg: boolean**
  curl E is returned at the target locations if ifcurlE = true
- **ifdivEtarg: boolean**
  div E is returned at the target locations if ifdivE = true

Returns:

- vals.E: E field defined above at target locations if requested $(E(t_j))$

- vals.curlE: curl of E field at target locations if requested $(\nabla \times E(t_j))$
- vals.divE: divergence of E at target locations if requested $(\nabla \cdot E(t_j))$
- vals.Etarg: E field defined above at target locations if requested $(E(t_j))$
- vals.curlEtarg: curl of E field at target locations if requested $(\nabla \times E(t_j))$
- vals.divEtarg: divergence of E at target locations if requested $(\nabla \cdot E(t_j))$

Wrapper for direct evaluation of Maxwell N-body interactions. Note that this wrapper only returns fields, divergences, and curls at the target locations.

```
vals = em3ddir(zk,sources,targets;h_current=nothing,e_current=nothing,e_charge=nothing,
            ifEtarg=false,ifdivEtarg=false,ifcurlEtarg=false,
            nd=1,thresh=1e-16)
```

Back to top

CHAPTER

SEVEN

FMMLIB3D LEGACY INTERFACES

The current version of the FMM codes are backward compatible with the previous version of this library: FMMLIB3D. On this page, we refer to these wrappers as the legacy wrappers.

> **Note**
>
> The field associated with the potential returned in FMMLIB3D is negative of the gradient of the potential.

- Laplace wrappers
- Helmholtz wrappers

7.1 Laplace

The legacy Fortran Laplace wrappers are contained in `src/Laplace/lfmm3dwrap_legacy.f` and the legacy MATLAB Laplace wrappers are contained in `matlab/lfmm3dpart.m` and `matlab/l3dpartdirect.m`.

Currently we have interfaces for the following four Fortran wrappers and two matlab wrappers:

- Two self evaluation wrappers (*lfmm3dpart and lfmm3dpartself*)
- The main fmm wrapper and direct evaluation wrapper in fortran (*lfmm3dparttarg and l3dpartdirect*)
- *MATLAB wrappers*

> **Note**
>
> In the Laplace wrappers for FMMLIB3D, the charge strengths, dipole strengths, potentials, and fields are complex numbers as opposed to real numbers for the rest of the library.

> **Note**
>
> lfmm3dpartself and lfmm3dpart are identical subroutines except for their names.

7.1.1 lfmm3dpart and lfmm3dpartself

- Evaluation points: Sources
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
subroutine lfmm3dpart(ier,iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,pot,iffld,fld)
```

```
subroutine lfmm3dpartself(ier,iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,
↪dipvec,ifpot,pot,iffld,fld)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right) \right)$$

at the source locations $x = x_j$. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

Input arguments:

- **iprec: integer**

  precision flag

  iprec=-2 => tolerance = 0.5d0

  iprec=-1 => tolerance = 0.5d-1

  iprec=0 => tolerance = 0.5d-2

  iprec=1 => tolerance = 0.5d-3

  iprec=2 => tolerance = 0.5d-6

  iprec=3 => tolerance = 0.5d-9

  iprec=4 => tolerance = 0.5d-12

- **nsource: integer**
  Number of sources

- **source: double precision(3,nsource)**
  Source locations, x_j

- **ifcharge: integer**

  charge computation flag

  ifcharge =1 => include charge contribution, otherwise do not

- **charge: double complex(nsource)**
  Charge strengths, c_j

- **ifdipole: integer**

  dipole computation flag

  ifdipole =1 => include dipole contribution, otherwise do not

- **dipstr: double complex(nsource)**
  Dipole strengths, d_j

- **dipvec: double precision(3,nsource)**
  Dipole orientation vectors, v_j

- **ifpot: integer**

  potential flag

  ifpot =1 => compute potential, otherwise do not

- **iffld: integer**

Field flag

iffld =1 => compute field, otherwise do not

Output arguments:

- **ier: integer**

  error code, currently unused

- **pot: double complex(nsource)**

  Potential at source locations, if requested, $u(x_j)$

- **fld: double complex(3,nsource)**

  Field at source locations, if requested, -$\nabla u(x_j)$

Back to Laplace legacy wrappers

Back to top

7.1.2 lfmm3dparttarg and l3dpartdirect

- Evaluation points: Sources/Targets/Sources+targets
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
subroutine lfmm3dparttarg(ier,iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,
↪dipvec,ifpot,pot,iffld,fld,ntarg,targ,ifpottarg,pottarg,iffldtarg,fldtarg)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|}\right)\right)$$

at the source locations $x = x_j$/target locations $x = t_j$/ source and target locations. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

Input arguments:

- **iprec: integer**

  precision flag

  iprec=-2 => tolerance = 0.5d0

  iprec=-1 => tolerance = 0.5d-1

  iprec=0 => tolerance = 0.5d-2

  iprec=1 => tolerance = 0.5d-3

  iprec=2 => tolerance = 0.5d-6

  iprec=3 => tolerance = 0.5d-9

  iprec=4 => tolerance = 0.5d-12

- **nsource: integer**

  Number of sources

- **source: double precision(3,nsource)**

  Source locations, x_j

- **ifcharge: integer**

  charge computation flag

ifcharge =1 => include charge contribution, otherwise do not

- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ifdipole: integer**

  dipole computation flag

  ifdipole =1 => include dipole contribution, otherwise do not
- **dipstr: double complex(nsource)**
  Dipole strengths, d_j
- **dipvec: double precision(3,nsource)**
  Dipole orientation vectors, v_j
- **ifpot: integer**

  potential flag

  ifpot =1 => compute potential, otherwise do not
- **iffld: integer**

  Field flag

  iffld =1 => compute field, otherwise do not
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Source locations, x_j
- **ifpottarg: integer**

  target potential flag

  ifpottarg =1 => compute potential, otherwise do not
- **iffldtarg: integer**

  target field flag

  iffldtarg =1 => compute field, otherwise do not

Output arguments:

- **ier: integer**
  error code, currently unused
- **pot: double complex(nsource)**
  Potential at source locations, if requested, $u(x_j)$
- **fld: double complex(3,nsource)**
  Field at source locations, if requested, -$\nabla u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, if requested, $u(t_j)$
- **fld: double complex(3,ntarg)**
  Field at source locations, if requested, -$\nabla u(t_j)$

Wrapper for direct evaluation of Laplace N-body interactions.

```
subroutine l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,pot,
↪iffld,fld,ntarg,targ,ifpottarg,pottarg,iffldtarg,fldtarg)
```

Example:

- see examples/lfmm3d_legacy_example.f. The corresponding makefile is examples/lfmm3d_legacy_example.make.

Back to Laplace legacy wrappers

Back to top

7.1.3 MATLAB wrappers

- *matlab/lfmm3dpart.m*
- Evaluation points: Sources/Targets/Sources+targets
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
function [U]=lfmm3dpart(iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|} \right) \right)$$

at the source locations $x = x_j$/target locations $x = t_j$/ source and target locations. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

See *lfmm3dparttarg and l3dpartdirect* for a detailed description of input and output arguments. The output pot,pottarg,fld,fldtarg are contained in the output structure U.

The function can be called in 4 different ways

```
function [U]=lfmm3dpart(iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec)
function [U]=lfmm3dpart(iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld)
function [U]=lfmm3dpart(iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ)
function [U]=lfmm3dpart(iprec,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

The default argument for ifpot,iffld,ifpottarg,iffldtarg is 1, the defaults for ntarg is 0, and targ is zeros(3,1)

Wrapper for direct evaluation of Laplace N-body interactions.

- *matlab/l3dpartdirect.m*

```
function [U]=l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,
↪iffld,ntarg,targ,ifpottarg,iffldtarg)
```

The function can be called in 4 different ways

```
function [U]=l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec)
function [U]=l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,
↪iffld)
function [U]=l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,
↪iffld,ntarg,targ)
function [U]=l3dpartdirect(nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,
↪iffld,ntarg,targ,ifpottarg,iffldtarg)
```

Example:

- see `matlab/test_lfmm3dpart_direct.m`.

Back to Laplace legacy wrappers

Back to top

7.2 Helmholtz

The legacy Fortran Helmholtz wrappers are contained in `src/Helmholtz/hfmm3dwrap_legacy.f` and the legacy MATLAB Helmholtz wrappers are contained in `matlab/hfmm3dpart.m` and `matlab/h3dpartdirect.m`.

Currently we have interfaces for the following four Fortran wrappers and two matlab wrappers:

- Two self evaluation wrappers (*hfmm3dpart and lfmm3dpartself*)
- The main fmm wrapper and direct evaluation wrapper in fortran (*hfmm3dparttarg and h3dpartdirect*)
- *MATLAB wrappers*

Note

hfmm3dpartself and hfmm3dpart are identical subroutines except for their names.

7.2.1 hfmm3dpart and lfmm3dpartself

- Evaluation points: Sources
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
subroutine hfmm3dpart(ier,iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,pot,iffld,fld)
```

```
subroutine hfmm3dpartself(ier,iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,
↪dipvec,ifpot,pot,iffld,fld)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|}\right)\right)$$

at the source locations $x = x_j$. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

Input arguments:

- **iprec: integer**

  precision flag

  iprec=-2 => tolerance = 0.5d0

  iprec=-1 => tolerance = 0.5d-1

  iprec=0 => tolerance = 0.5d-2

  iprec=1 => tolerance = 0.5d-3

  iprec=2 => tolerance = 0.5d-6

  iprec=3 => tolerance = 0.5d-9

  iprec=4 => tolerance = 0.5d-12

- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **ifcharge: integer**

  charge computation flag

  ifcharge =1 => include charge contribution, otherwise do not

- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ifdipole: integer**

  dipole computation flag

  ifdipole =1 => include dipole contribution, otherwise do not

- **dipstr: double complex(nsource)**
  Dipole strengths, d_j
- **dipvec: double precision(3,nsource)**
  Dipole orientation vectors, v_j
- **ifpot: integer**

  potential flag

  ifpot =1 => compute potential, otherwise do not

- **iffld: integer**

  Field flag

  iffld =1 => compute field, otherwise do not

Output arguments:

- **ier: integer**
  error code, currently unused
- **pot: double complex(nsource)**
  Potential at source locations, if requested, $u(x_j)$
- **fld: double complex(3,nsource)**
  Field at source locations, if requested, -$\nabla u(x_j)$

Back to Helmholtz legacy wrappers

Back to top

7.2.2 hfmm3dparttarg and h3dpartdirect

- Evaluation points: Sources/Targets/Sources+targets
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
subroutine hfmm3dparttarg(ier,iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,
↪dipvec,ifpot,pot,iffld,fld,ntarg,targ,ifpottarg,pottarg,iffldtarg,fldtarg)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{e^{ik\|x-x_j\|}}{4\pi\|x-x_j\|} \right)\right)$$

at the source locations $x = x_j$/target locations $x = t_j$/ source and target locations. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

Input arguments:

- **iprec: integer**

  precision flag

  iprec=-2 => tolerance = 0.5d0

  iprec=-1 => tolerance = 0.5d-1

  iprec=0 => tolerance = 0.5d-2

  iprec=1 => tolerance = 0.5d-3

  iprec=2 => tolerance = 0.5d-6

  iprec=3 => tolerance = 0.5d-9

  iprec=4 => tolerance = 0.5d-12

- **zk: double complex**
  Helmholtz parameter, k
- **nsource: integer**
  Number of sources
- **source: double precision(3,nsource)**
  Source locations, x_j
- **ifcharge: integer**

  charge computation flag

  ifcharge =1 => include charge contribution, otherwise do not

- **charge: double complex(nsource)**
  Charge strengths, c_j
- **ifdipole: integer**

  dipole computation flag

  ifdipole =1 => include dipole contribution, otherwise do not

- **dipstr: double complex(nsource)**
  Dipole strengths, d_j
- **dipvec: double precision(3,nsource)**
  Dipole orientation vectors, v_j
- **ifpot: integer**

  potential flag

  ifpot =1 => compute potential, otherwise do not
- **iffld: integer**

  Field flag

  iffld =1 => compute field, otherwise do not
- **ntarg: integer**
  Number of targets
- **targ: double precision(3,ntarg)**
  Source locations, x_j
- **ifpottarg: integer**

  target potential flag

  ifpottarg =1 => compute potential, otherwise do not
- **iffldtarg: integer**

  target field flag

  iffldtarg =1 => compute field, otherwise do not

Output arguments:

- **ier: integer**
  error code, currently unused
- **pot: double complex(nsource)**
  Potential at source locations, if requested, $u(x_j)$
- **fld: double complex(3,nsource)**
  Field at source locations, if requested, -$\nabla u(x_j)$
- **pottarg: double complex(ntarg)**
  Potential at target locations, if requested, $u(t_j)$
- **fld: double complex(3,ntarg)**
  Field at source locations, if requested, -$\nabla u(t_j)$

Wrapper for direct evaluation of Helmholtz N-body interactions.

```
subroutine h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,ifpot,
↪pot,iffld,fld,ntarg,targ,ifpottarg,pottarg,iffldtarg,fldtarg)
```

Example:

- see `examples/hfmm3d_legacy_example.f`. The corresponding makefile is `examples/hfmm3d_legacy_example.make`.

Back to Helmholtz legacy wrappers

Back to top

7.2.3 MATLAB wrappers

- *matlab/hfmm3dpart.m*
- Evaluation points: Sources/Targets/Sources+targets
- Interaction kernel: Charges/Dipoles/Charges+Dipoles
- Outputs requested: Potential/Fields/Potential+Fields

```
function [U]=hfmm3dpart(iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

This subroutine evaluates the potential/field/potential and field

$$u(x) = \sum_{j=1}^{N} c_j \frac{1}{4\pi\|x - x_j\|} - d_j \left(v_j \cdot \nabla \left(\frac{1}{4\pi\|x - x_j\|}\right)\right)$$

at the source locations $x = x_j$/target locations $x = t_j$/ source and target locations. When $x = x_m$, the term corresponding to x_m is dropped from the sum.

See *hfmm3dparttarg and h3dpartdirect* for a detailed description of input and output arguments. The output pot,pottarg,fld,fldtarg are contained in the output structure U.

The function can be called in 4 different ways

```
function [U]=hfmm3dpart(iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec)
function [U]=hfmm3dpart(iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld)
function [U]=hfmm3dpart(iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ)
function [U]=hfmm3dpart(iprec,zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

The default argument for ifpot,iffld,ifpottarg,iffldtarg is 1, the defaults for ntarg is 0, and targ is zeros(3,1)

Wrapper for direct evaluation of Helmholtz N-body interactions.

- *matlab/h3dpartdirect.m*

```
function [U]=h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

The function can be called in 4 different ways

```
function [U]=h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec)
function [U]=h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld)
function [U]=h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ)
function [U]=h3dpartdirect(zk,nsource,source,ifcharge,charge,ifdipole,dipstr,dipvec,
↪ifpot,iffld,ntarg,targ,ifpottarg,iffldtarg)
```

Example:

- see `matlab/test_hfmm3dpart_direct.m`.

Back to Helmholtz legacy wrappers

Back to top

CHAPTER

EIGHT

ACKNOWLEDGMENTS

Principal contributors to this code base are Travis Askham, Zydrunas Gimbutas, Leslie Greengard, Libin Lu, Jeremy Magland, Dhairya Malhotra, Michael O'Neil, Manas Rachh, Vladimir Rokhlin, and Felipe Vico.

Thanks to Guilherme Saturnino for contributing workflows for continuous integration and for making the library a pip installable package.

Thanks to Joakim Anden, Alex Barnett, Shidong Jiang, David Stein, and Jun Wang for several useful discussions.

CHAPTER

NINE

REFERENCES

References for this software and the underlying mathematics include:

BIBLIOGRAPHY

[1] Hongwei Cheng, Leslie Greengard, and Vladimir Rokhlin. A fast adaptive multipole algorithm in three dimensions. *Journal of computational physics*, 155(2):468–498, 1999.

[2] Leslie Greengard, Jingfang Huang, Vladimir Rokhlin, and Stephen Wandzura. Accelerating fast multipole methods for the helmholtz equation at low frequencies. *IEEE Computational Science and Engineering*, 5(3):32–38, 1998.

[3] Leslie Greengard and Vladimir Rokhlin. A new version of the fast multipole method for the laplace equation in three dimensions. *Acta numerica*, 6:229–269, 1997.

[4] Leslie F Greengard and Jingfang Huang. A new version of the fast multipole method for screened coulomb interactions in three dimensions. *Journal of Computational Physics*, 180(2):642–658, 2002.