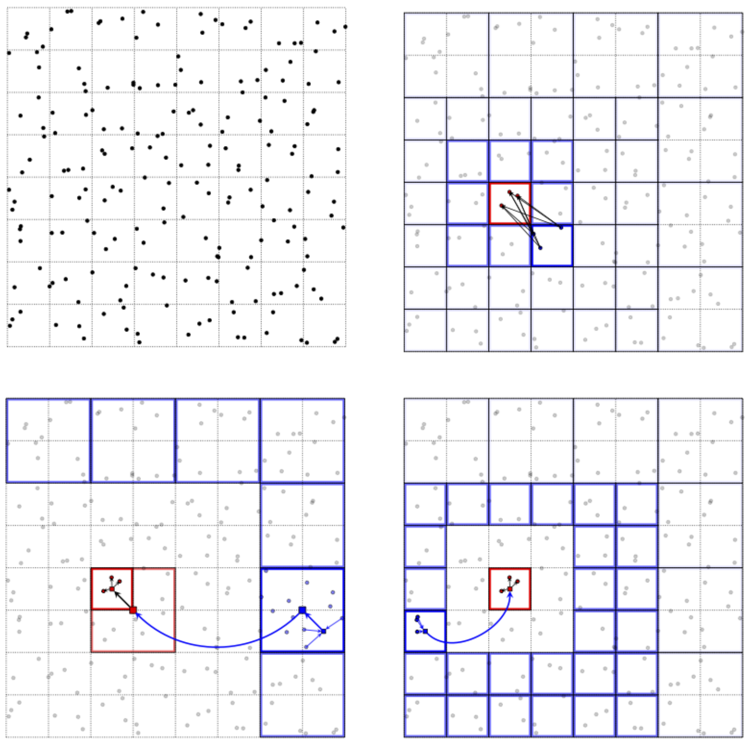
fmm3d Documentation

Release 1.0.0

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

May 01, 2020

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

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

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` and `pip3`
- for building standard MATLAB wrappers: MATLAB
- for modifying MATLAB wrappers (experts only): `mwrap`

1.3 Quick install instructions

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

- For linux, run `make test`.
- For Mac OSX, run `cp make.inc.macosx_gcc make.inc` followed by `make test`.
- For Windows, run `copy make.inc.windows make.inc` followed by `mingw32-make test`

This should compile the static library in `lib-static/` and some fortran test drivers in `test/`, after which it runs the test programs. The last 10 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 18 out of 18 tests in hfmm3d vec 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
```

(continues on next page)

(continued from previous page)

```
Successfully completed 18 out of 18 tests in lfmm3d vec testing suite
rm print_testreshelm.txt
rm print_testreslap.txt
```

Note: By default, `make test` 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 test` should be replaced by `make test FAST_KER=ON`. See *Custom library compilation options* for other options.

If `make test` fails, see more detailed instructions below. If it succeeds, run `make lib`, which creates the dynamic library (`libfmm3d.so`). You may then link to the FMM library using the `-lfmm3d` option.

Note: On MacOSX, in order to link with the dynamic libraries, you will need to copy libfmm3d.so to `usr/local/lib`. See any of the makefiles in the `examples/` directory for prototypes.

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 or multi-threading.

In order to enable multi-threading, append `OMP=ON` 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.

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 python3 and pip3 installed.

You may then execute `make python3` (after copying over the operating system specific make.inc.* file to make.inc) which calls pip3 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.

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`)

1.5 Building the MATLAB wrappers

First make sure you have MATLAB installed.

The library comes with precompiled interfaces of the easy-to-install version of the library and can be directly called from MATLAB. However, we **strongly** recommend compiling the mex interfaces on your machine.

This can be done using `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`.

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
```

1.7 Tips for installing optional dependencies

1.7.1 Installing python3 and pip3

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
```

Then use *make python3* instead of *make python*. You will only need to do this in case the default version of *python* and *pip* is not >=3.0

1.7.2 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 or later from http://www.cs.cornell.edu/~bindel/sw/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

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

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

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

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

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

at the source and target locations.

Similarly, for the vectorized Helmholtz FMM, let $c_{\ell,j} \in \mathbb{C}$, $j = 1,2,\dots N$, $\ell = 1,2,\dots n_d$ 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

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

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

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

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 $c_j/\|x - x_j\|$, and the dipole interactions are given by $-v_j \cdot \nabla(1/\|x - x_j\|)$
  - 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/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*)
  - Interaction Type: Dipoles
    - Potential (*lfmm3d_s_d_p*)
    - Gradient (*lfmm3d_s_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*lfmm3d_s_cd_p*)
    - Gradient (*lfmm3d_s_cd_g*)
- Evaluation points: Targets
  - Interaction Type: Charges
    - Potential (*lfmm3d_t_c_p*)
    - Gradient (*lfmm3d_t_c_g*)
  - Interaction Type: Dipoles
    - Potential (*lfmm3d_t_d_p*)
    - Gradient (*lfmm3d_t_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*lfmm3d_t_cd_p*)

- Gradient (*lfmm3d_t_cd_g*)

- Evaluation points: Sources + Targets
  - Interaction Type: Charges
    - Potential (*lfmm3d_st_c_p*)
    - Gradient (*lfmm3d_st_c_g*)
  - Interaction Type: Dipoles
    - Potential (*lfmm3d_st_d_p*)
    - Gradient (*lfmm3d_st_d_g*)
  - Interaction Type: Charges + Dipoles
    - Potential (*lfmm3d_st_cd_p*)
    - Gradient (*lfmm3d_st_cd_g*)

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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,pottarg,
↪gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,pottarg,
↪gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,
↪pottarg,gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

Back to Laplace FMM

Back to top

3.2 Helmholtz FMM

The Helmholtz FMM evaluates the following potential and its gradient

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

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 $c_j/\|x-x_j\|$, and the dipole interactions are given by $-v_j \cdot \nabla(1/\|x-x_j\|)$
  - 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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(x_j)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(x_j)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(x_j)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,pottarg,
↪gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,pottarg,
↪gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

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,ntarg,targ,pot,pottarg)
```

This subroutine evaluates the potential

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

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, $u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential

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

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, $u_\ell(t_i)$

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,ntarg,targ,pot,grad,
↪pottarg,gradtarg)
```

This subroutine evaluates the potential and its gradient

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

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, $\nabla u(t_i)$

Vectorized version:

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

This subroutine evaluates the potential and its gradient

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

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, $\nabla u_\ell(t_i)$

Back to Helmholtz FMM

Back to top

3.3 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`

Back to top

CHAPTER

FOUR

MATLAB

The MATLAB interface has four callable subroutines:

- Helmholtz wrappers: Fast multipole implementation (hfmm3d) and direct evaluation (h3ddir) for Helmholtz N-body interactions
- Laplace wrappers: Fast multipole implementation (lfmm3d) and direct evaluation (l3ddir) for Laplace N-body interactions

4.1 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

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

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 `hfmmexample.m`

Back to top

4.2 Laplace wrappers

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

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

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 `lfmmexample.m`

Back to top

CHAPTER FIVE

PYTHON

The Python interface has four callable subroutines:

- Helmholtz wrappers: Fast multipole implementation (hfmm3d) and direct evaluation (h3ddir) for Helmholtz N-body interactions
- Laplace wrappers: Fast multipole implementation (lfmm3d) and direct evaluation (l3ddir) for Laplace N-body interactions

5.1 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

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

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 `hfmmexample.py`

Back to top

5.2 Laplace wrappers

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

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

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 `lfmmexample.py`

Back to top

CHAPTER

SIX

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

6.1 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.

6.1.1 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

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

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

6.1.2 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

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

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

6.1.3 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

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

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

6.2 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.

6.2.1 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

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

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

6.2.2 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

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

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

6.2.3 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

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

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 SEVEN

ACKNOWLEDGMENTS

Principal contributors to this code base are Zydrunas Gimbutas, Leslie Greengard, Libin Lu, Jeremy Magland, Dhairya Malhotra, Michael O'Neil, Manas Rachh, and Vladimir Rokhlin

Thanks to Joakim Anden, Travis Askham, Alex Barnett, David Stein, Felipe Vico, and Jun Wang for several useful discussions

CHAPTER EIGHT

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.