# FMMLIB3D

Version 1.2

FORTRAN 90/95

April, 2012

User's guide

# Contents

# 1 Introduction

This manual describes the use of the FMMLIB3D suite for the evaluation of potential fields, governed by either the Laplace or Helmholtz equation in free space. The codes are easy to use and reasonably well optimized for performance on either single core processors, or small multi-core systems using OpenMP. FMMLIB3D is being released under the terms of the GNU General Public License (version 2), as published by the Free Software Foundation.

The fast multipole method (FMM) computes N-body interactions in approximately linear time for non-pathological particle distributions, assuming in the case of the Helmholtz equation that the entire computational domain is a modest number of wavelengths in size. This is the "low frequency" regime from the point of view of either scattering theory or FMM implementations. (The high-frequency version of the FMM requires a more complex algorithm, and has not been incorporated into this software. We do, however, provide a subroutine which evaluates the scattered field at some distance from the scatterer, using a single multipole expansion about the center of the scatterer.) More precisely, FMMLIB3D computes sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j \, G_k(\mathbf{y}_i - \mathbf{x}_j) + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} G_k(\mathbf{y}_i - \mathbf{x}_j)$$

for $i = 1, \ldots, N$, where

$$G_k(\mathbf{x}) = \frac{e^{ik\|\mathbf{x}\|}}{\|\mathbf{x}\|}$$

$q_j$ is referred to as the charge strength and $p_j$ as the dipole strength. $\mathbf{n} = (n_1, n_2, n_3)$ is a vector whose direction determines the dipole orientation (if present).

When $k = 0$, $G_k$ is the Green's function (up to scaling by $\frac{1}{4\pi}$) for the Laplace equation. For $k \neq 0$, we assume that $k$ is in the upper half of the complex plane with $\Re(k) \geq \Im(k)/10$. It is designed for scattering calculations, and there are scaling issues that need to be incorporated to handle the modified Helmholtz (Yukawa) regime where $k$ is nearer the imaginary axis. (This will be fixed in a forthcoming code release.)

---

**Important note:** The charge and dipole strengths are assumed to be **complex** double precision numbers for both the Laplace and Helmholtz libraries. If you pass a **real** array, the code will not execute correctly.

---

*This package provides a fully adaptive version of the FMM for the research community. It is not the most highly optimized version possible, intended rather to be accessible and modifiable with only modest effort. The translation operators used in FMMLIB3D are based on rotation and translation along the z-axis. For a fully optimized code, plane wave-based operators should be used [1, 2]. This, however, would add significant complexity to the code, and would make the algorithm less transparent to the user and harder to modify.*

*The internal documentation of lower level routines is mixed, but this is a work in progress. The higher level routines (we hope) should be clear.*

## 1.1 Subroutine LFMM3DPARTSELF

---

subroutine lfmm3dpartself(ier, iprec, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld)

computes sums of the form

$$\phi(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j\neq i}}^{N} q_j \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} \left( \frac{1}{\|\mathbf{x}_i - \mathbf{x}_j\|} \right) \tag{1}$$

for $i = 1, \ldots, N$, as well as derivatives of $\phi$.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-14}$. |

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (1)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` $= 1$, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (1)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

---

**Note that the charge, dipstr, pot, fld arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.2 Subroutine LFMM3DPARTTARG

---

subroutine lfmm3dparttarg(ier, iprec, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j \frac{1}{\|\mathbf{y}_i - \mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} \left( \frac{1}{\|\mathbf{y}_i - \mathbf{x}_j\|} \right)$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (1) if desired.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-15}$. |

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (1)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` $= 1$, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (1)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :

fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg, arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.3 Subroutine L3DPARTDIRECT

---

subroutine l3dpartdirect(nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j \frac{1}{\|\mathbf{y}_i - \mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} \left( \frac{1}{\|\mathbf{y}_i - \mathbf{x}_j\|} \right)$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (1) if desired. **It implements the summation formula directly and is not fast.**

---

**Input Parameters:**

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (1)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (1)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.4 Subroutine LFMM3DTRIASELF

---

subroutine lfmm3dtriaself(ier, iprec, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld)

computes sums of the form

$$\phi(\mathbf{x}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{1}{\|\mathbf{x}_i - \mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{1}{\|\mathbf{x}_i - \mathbf{x}\|} \right) d\mathbf{x} \tag{2}$$

for $i = 1, \ldots, N$, as well as derivatives of $\phi$.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-14}$. |

`nsource` ***integer*** :
number of triangles

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (2)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (2)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the principal value of the potential at the ith centroid (source).

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamard finite part) of the field (-gradient of the potential) at the ith centroid (source).

---

**Note that the charge, dipstr, pot, fld arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.5 Subroutine LFMM3DTRIATARG

---

subroutine lfmm3dtriatarg(ier, iprec, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{1}{\|\mathbf{y}_i - \mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{1}{\|\mathbf{y}_i - \mathbf{x}\|} \right) d\mathbf{x}$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (2) if desired.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-15}$. |

`nsource` ***integer*** :
number of sources

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (2)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (2)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the principal value of the potential at the ith centroid (source).

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamard finite part) of the field (-gradient of the potential) at the ith centroid (source).

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg, arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.6 Subroutine L3DTRIADIRECT

---

subroutine l3dtriadirect(nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{1}{\|\mathbf{y}_i - \mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{1}{\|\mathbf{y}_i - \mathbf{x}\|} \right) d\mathbf{x}$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (2) if desired. **It implements the summation formula directly and is not fast.**

---

**Input Parameters:**

`nsource` ***integer*** :
number of sources

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (2)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (2)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth

triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`pot(nsources)` ***complex *16*** :
pot(i) is the principal value of the potential at the ith centroid (source).

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamard finite part) of the field (-gradient of the potential) at the ith centroid (source).

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.7 Subroutine HFMM3DPARTSELF

---

subroutine hfmm3dpartself(ier, iprec, zk, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld)

compute sums of the form

$$\phi(\mathbf{x}_i) = \sum_{\substack{j=1 \\ j\neq i}}^{N} q_j \frac{e^{ik\|x_i-\mathbf{x}_j\|}}{\|\mathbf{x}_i-\mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j}\left(\frac{e^{ik\|x_i-\mathbf{x}_j\|}}{\|\mathbf{x}_i-\mathbf{x}_j\|}\right) \tag{3}$$

for $i = 1, \ldots, N$.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-15}$. |

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (3)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` $= 1$, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (3)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

---

**Note that the charge, dipstr, pot, fld arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.8 Subroutine HFMM3DPARTTARG

---

subroutine hfmm3dparttarg(ier, iprec, zk, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}_j\|}}{\|\mathbf{y}_i - \mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} \left( \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}_j\|}}{\|\mathbf{y}_i - \mathbf{x}_j\|} \right)$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (3) if desired.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |
| `iprec` $= 2$ | for least squares errors $< 0.5\,10^{-6}$. |
| `iprec` $= 3$ | for least squares errors $< 0.5\,10^{-9}$. |
| `iprec` $= 4$ | for least squares errors $< 0.5\,10^{-12}$. |
| `iprec` $= 5$ | for least squares errors $< 0.5\,10^{-15}$. |

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (1)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` $= 1$, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (1)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :

fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.9 Subroutine H3DPARTDIRECT

---

subroutine h3dpartdirect(zk, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} q_j \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}_j\|}}{\|\mathbf{y}_i - \mathbf{x}_j\|} + p_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}_j} \left( \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}_j\|}}{\|\mathbf{y}_i - \mathbf{x}_j\|} \right)$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (3) if desired. **It implements the summation formula directly and is not fast.**

---

**Input Parameters:**

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the jth source in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of the charge sources. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the jth charge ($q_j$ in the formula (1)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of the dipole sources. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the strength of the jth dipole ($p_j$ in the formula (1)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\mathbf{n}_j$ in the formula (1)).

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`pot(nsources)` ***complex *16*** :
pot(i) is the potential at the ith source

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the field (-gradient of the potential) at the ith source

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.10 Subroutines HFMM3DTRIASELF

---

subroutine hfmm3dtriaself(ier, iprec, zk, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld)

compute sums of the form

$$\phi(\mathbf{x}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{e^{ik\|x_i-\mathbf{x}\|}}{\|\mathbf{x}_i-\mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{e^{ik\|\mathbf{x}_i-\mathbf{x}\|}}{\|\mathbf{x}_i-\mathbf{x}\|} \right) d\mathbf{x} \tag{4}$$

for $i = 1, \ldots, N$.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |

This flag controls the FMM error, but not the error in the nearest neighbor integrals. For those, we subtract the dominant singularity (computing the integral with the $1/r$ kernel exactly), and use numerical quadrature for the difference between the Helmholtz and Laplace kernels ($(e^{ikr}-1)/r$, etc.). As iprec is set to higher values, we use more and more points in these numerical quadratures but the accuracy is not guaranteed for highly irregular triangulations.

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` $= 1$, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (4)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (4)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the principal part of the potential at the ith centroid (source).

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamard finie part) of the field (-gradient of the potential) at the ith centroid (source).

---

**Note that the charge, dipstr, pot, fld arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.11 Subroutine HFMM3DTRIATARG

---

subroutine hfmm3dtriatarg(ier, iprec, zk, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|} \right) d\mathbf{x}$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (4) if desired.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

$$\begin{aligned}
\texttt{iprec} &= -2 \quad \text{for least squares errors} < 0.5\,10^{0},\\
\texttt{iprec} &= -1 \quad \text{for least squares errors} < 0.5\,10^{-1},\\
\texttt{iprec} &= 0 \quad \text{for least squares errors} < 0.5\,10^{-2},\\
\texttt{iprec} &= 1 \quad \text{for least squares errors} < 0.5\,10^{-3}.
\end{aligned}$$

This flag controls the FMM error, but not the error in the nearest neighbor integrals. For those, we subtract the dominant singularity (computing the integral with the $1/r$ kernel exactly), and use numerical quadrature for the difference between the Helmholtz and Laplace kernels ($(e^{ikr}-1)/r$, etc.). As iprec is set to higher values, we use more and more points in these numerical quadratures but the accuracy is not guaranteed for highly irregular triangulations.

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (4)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (4)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pot(nsources)` ***complex *16*** :
pot(i) is the principal value of the potential at the ith centroid (source)

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamard finite part) of the field (-gradient of the potential) at the ith centroid (source)

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.12 Subroutine HFMM3DTRIAMPFTARG

---

subroutine hfmm3dtriampftarg(ier, iprec, zk, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{e^{ik\|\mathbf{y}_i-\mathbf{x}\|}}{\|\mathbf{y}_i-\mathbf{x}\|} \right) d\mathbf{x}$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (4) if desired. Unlike hfmm3dtriatarg, it is capable of handling distant targets. It would typically be used after solving an integral equation in order to compute a far field signature.

---

**Input Parameters:**

`iprec` ***integer*** :
precision flag. Allowed values are

| | |
|---|---|
| `iprec` $= -2$ | for least squares errors $< 0.5\,10^{0}$, |
| `iprec` $= -1$ | for least squares errors $< 0.5\,10^{-1}$, |
| `iprec` $= 0$ | for least squares errors $< 0.5\,10^{-2}$, |
| `iprec` $= 1$ | for least squares errors $< 0.5\,10^{-3}$. |

This flag controls the FMM error, but not the error in the nearest neighbor integrals. For those, we subtract the dominant singularity (computing the integral with the $1/r$ kernel exactly), and use numerical quadrature for the difference between the Helmholtz and Laplace kernels ($(e^{ikr}-1)/r$, etc.). As iprec is set to higher values, we use more and more points in these numerical quadratures but the accuracy is not guaranteed for highly irregular triangulations.

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of sources

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (4)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (4)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`ier` ***integer*** :
Error return codes.
`ier = 0`: Successful completion of code.
`ier = 4`: failure to allocate memory for oct-tree
`ier = 8`: failure to allocate memory for FMM workspaces
`ier = 16`: failure to allocate meory for multipole/local expansions

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

## 1.13 Subroutine H3DTRIADIRECT

---

subroutine h3dtriadirect(nqtri, zk, nsource, triaflat, trianorm, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifpot, pot, iffld, fld, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg)

compute sums of the form

$$\phi(\mathbf{y}_i) = \sum_{j=1}^{N} \int_{T_j} \sigma_j \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}\|}}{\|\mathbf{y}_i - \mathbf{x}\|} + \mu_j \mathbf{n}_j \cdot \nabla_{\mathbf{x}} \left( \frac{e^{ik\|\mathbf{y}_i - \mathbf{x}\|}}{\|\mathbf{y}_i - \mathbf{x}\|} \right) d\mathbf{x}$$

for $i = 1, \ldots, N_t$, as well as derivatives of $\phi$. It also returns sums of the form (4) if desired. **It implements the summation formula directly and is not fast.**

---

**Input Parameters:**

`nqtri` ***integer*** :
number of quadrature nodes on the triangle used to compute the integral on each triangle after singularity subtraction - that is, the difference between the Laplace and Helmholtz kernels. The code is presently not robust with respect to this parameter. `nqtri` = 6 yields about 3 digits. `nqtri` = 12 yields about 6 digits.

`zk` ***complex *16*** :
Helmholtz parameter

`nsource` ***integer*** :
number of triangles (sources).

`triaflat(3,3,nsources)` ***real *8*** :
triaflat(i,k,j) is the ith component of the kth vertex of the jth triangle in $\mathbf{R}^3$. The triangles are assumed to be positively oriented with respect to the vertex ordering. (That is, the outward normal follows the right-hand rule.)

`trianorm(3,nsources)` ***real *8*** :
trianorm(k,j) is the kth component of the normal on the jth triangle in $\mathbf{R}^3$.

`source(3,nsources)` ***real *8*** :
sources(k,j) is the kth component of the centroid of the jth triangle in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `icharge` = 1, then include the effect of a single layer potential with piecewise constant density given by the charge array. Otherwise, omit.

`charge(nsources)` ***complex *16*** :
charge(j) is the strength of the single layer potential on the jth triangle ($sigma_j$ in the formula (4)).

`ifdipole` ***integer*** :
dipole flag. If `idipole` = 1, then include the effect of a double layer potential with piecewise constant density given by the dipstr array. Otherwise, omit.

`dipstr(nsources)` ***complex *16*** :
dipstr(j) is the orientation of the dipole density on the jth triangle ($mu_j$ in the formula (4)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the dipole on the jth triangle. In the present code, this must match the trianorm array. In future releases, the dipole orientation will be permitted to be arbitrary.

`ifpot` ***integer*** :
potential flag. If `ifpot` = 1, the principal value of the potential is computed. Otherwise, it is not.

`iffld` ***integer*** :
field (gradient) flag. If `iffld` = 1 the principal part (or Hadamard finite part) of the gradient of the potential is computed. Otherwise, it is not.

`ntarget` ***integer*** :
number of targets

`target(3,ntarget)` ***real *8*** :
target(k,j) is the kth component of the jth target in $\mathbf{R}^3$.

`ifpottarg` ***integer*** :
target potential flag. If `ifpottarg` = 1, the potential is computed. Otherwise, it is not.

`iffldtarg` ***integer*** :
target field (gradient) flag. If `iffldtarg` = 1 the gradient of the potential is computed. Otherwise, it is not.

**Unused arrays do not need to be allocated in full. Thus, if ifcharge = 0, charge can be dimensioned as a (complex) scalar. If ifdipole = 0, dipstr can be dimensioned as a complex scalar and dipvec can be dimensioned in the calling program as dipvec(3) - BUT NOT dipvec(1).**

**Output Parameters:**

`pot(nsources)` ***complex *16*** :
pot(i) is the principal part of the potential at the ith centroid (source)

`fld(3,nsources)` ***complex *16*** :
fld(k,i) is the kth component of the principal value (or Hadamrd finite part) of the field (-gradient of the potential) at the ith source

`pottarg(ntarget)` ***complex *16*** :
pottarg(i) is the potential at the ith target

`fldtarg(3,ntarget)` ***complex *16*** :
fldtarg(k,i) is the kth component of the field (-gradient of the potential) at the ith target

---

**Note that the charge, dipstr, pot, fld, pottarg, fldtarg arrays must be declared and passed as complex arrays (even if the charge and dipole strengths are real).**

---

# 2 Sample drivers for FMMLIB3D

In the FMM3D/examples directory, the file `lfmm3dpart_driver.f` contains a sample driver for
`lfmm3dparttarg`. It creates a random distribution of source points on the unit sphere centered at the origin a random distribution of target points on a separated unit sphere, centered at $(0, 0, 2)$. The code then computes the potential and field at all source and target points. On a single core, with 10,000 sources, 10,000 targets, and iprec=1, the execution time should be one or two seconds.

The file `hfmm3dpart_driver.f` contains a sample driver for `hfmm3dparttarg`. It creates the same distribution of sources and targets, and sets the Helmholtz parameter to $k = 1 + 0.1i$. On a single core, with 10,000 sources, 10,000 targets, and iprec=1, the execution time should be about four seconds.

The file `lfmm3dtria_driver.f` contains a sample driver for `lfmm3dtriatarg`. On a single core, with 2,880 source triangles, 2,880 targets, and iprec=1, the execution time should be about three seconds.

The file `hfmm3dtria_driver.f` contains a sample driver for `hfmm3dtriatarg` and `hfmm3dtriampftarg`. On a single core, with 2,880 source triangles, 2,880 targets, and iprec=1, the execution time should be about five seconds.

## 3 Acknowledgments

This work was supported by the Department of Energy under contract DE-FGO288-ER-25053, by the Air Force Office of Scientific Research under MURI grant FA9550-06-1-0337 and NSSEFF Program Award FA9550-10-1-0180, by the National Science Foundation under grant DMS09-34733, and by a research grant from Meyer Sound Laboratories, Inc.

## References

[1] H. Cheng, L. Greengard and V. Rokhlin, "A Fast Adaptive Multipole Algorithm in Three Dimensions," *J. Comput. Phys.*, **155** pp.468-498, 1999.

[2] L. Greengard and V. Rokhlin, "A New Version of the Fast Multipole Method for the Laplace Equation in Three Dimensions," *Acta Numerica*, pp. 229-269, 1997.