StokesLib3D

Version 1.2

FORTRAN 90/95

May, 2012

User's guide

Contents

1 Introduction

This manual describes the use of the StokesLib3D suite for the evaluation of potential fields, governed by the Stokes 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. StokesLib3D is being released under the terms of the GNU General Public License (version 2), as published by the Free Software Foundation.

2 Stokes FMM

The flow of an incompressible Newtonian fluid at small values of Reynolds number Re is governed by the Stokes equation

$$\mu\Delta\mathbf{u} = \nabla p, \quad \operatorname{div}\mathbf{u} = 0, \tag{1}$$

where $\mathbf{u}$ is the velocity of the fluid, p is the pressure, and μ is the dynamic viscosity. Without loss of generality, we can set the value of viscosity $\mu = 1$ and consider the normalized version of the Stokes equation

$$\Delta\mathbf{u} = \nabla p, \quad \operatorname{div}\mathbf{u} = 0. \tag{2}$$

The standard fundamental solutions (Green's functions) for the Stokes flow are usually referred to as the stokeslet $\mathbf{S}$ and stresslet $\mathbf{T}$, respectively, which correspond to singular point forces and dipoles embedded in the flow. The fast multipole method (FMM) computes such Stokes N-body interactions in approximately linear time for non-pathological particle distributions. Informally, StokesLib3D computes sums of the form

$$\begin{aligned}
\mathbf{u}(\mathbf{y}^m) &= 4\pi \sum_{n=1}^{N} \mathbf{S}(\mathbf{y}^m - \mathbf{x}^n)\, \mathbf{f}^n + \mathbf{T}(\mathbf{y}^m - \mathbf{x}^n)\, \boldsymbol{\nu}^n\, \mathbf{g}^n, \\
p(\mathbf{y}^m) &= 4\pi \sum_{n=1}^{N} \mathbf{P}(\mathbf{y}^m - \mathbf{x}^n)\, \mathbf{f}^n + \boldsymbol{\Pi}(\mathbf{y}^m - \mathbf{x}^n)\, \boldsymbol{\nu}^n\, \mathbf{g}^n,
\end{aligned}$$

for $m = 1, \ldots, M$, where

$$S_{ij}(\mathbf{x}) = \frac{1}{8\pi}\left(\delta_{ij}\frac{1}{||\mathbf{x}||} + \frac{\mathbf{x}_i\mathbf{x}_j}{||\mathbf{x}||^3}\right), \quad T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \tag{3}$$

$$P_j = \frac{2}{8\pi}\frac{\mathbf{x}_j}{||\mathbf{x}||^3}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{4}$$

$\mathbf{f}^n$ are referred to as the charge strengths and $\mathbf{g}^n$ as the dipole strengths, and $\boldsymbol{\nu}^n$ are the vectors whose directions determine the dipole orientations (if present). More precisely, the sums computed by StokesFMMLib3D are of the form

$$\begin{aligned}
u_i(\mathbf{y}^m) &= \sum_{n=1}^{N}\sum_{j=1}^{3} S_{ij}(\mathbf{y}^m - \mathbf{x}^n)\, f_j^m + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T_{ijk}(\mathbf{y}^m - \mathbf{x}^n)\, \nu_k^n\, g_j^n, \\
p(\mathbf{y}^m) &= \sum_{n=1}^{N}\sum_{j=1}^{3} P_j(\mathbf{y}^m - \mathbf{x}^n)\, f_j^n + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi_{jk}(\mathbf{y}^m - \mathbf{x}^n)\, \nu_k^n\, g_j^n,
\end{aligned}$$

and we omit the normalization factor $\frac{1}{4\pi}$ from all sums.

In some situations, it is more convenient to consider modifications of the standard stresslet $\mathbf{T}$ and the corresponding pressure tensor $\mathbf{\Pi}$:

$$T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{5}$$

$$T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0, \tag{6}$$

$$T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right), \tag{7}$$

that are referred in the literature to as the stresslet (type 2), the rotlet, and the Stokes doublet, respectively. The StokesFMMLib3D routines are able to replace the standard stresslet $\mathbf{T}$ with one of these kernels with a help of a properly set flag.

In addition, StokesFMMLib3D is able to evaluate gradients of the velocities $\mathbf{u}$. Formulas for the strain $\boldsymbol{\varepsilon}$ and stress $\boldsymbol{\sigma}$ tensors, can then be obtained from partial derivatives of the preceding formulas for velocities with respect to each component x_i:

$$\varepsilon_{ij} = \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{8}$$

$$\sigma_{ij} = -\delta_{ij}p + \left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right), \tag{9}$$

and the vorticity $\boldsymbol{\omega}$ and the stress $\mathbf{t}$ vectors are

$$\boldsymbol{\omega} = \nabla \times \mathbf{u}, \quad \mathbf{t} = \boldsymbol{\sigma} \cdot \mathbf{n}, \tag{10}$$

where $\mathbf{n}$ is a unit direction vector.

3 Laplace FMM

In addition to the standard fundamental solutions to the Stokes equations, it is convenient to be able to evaluate the various harmonic contributions. The calculations of this type arise, for example, in evaluating pressure gradients ($\Delta\mathbf{S} = \nabla\mathbf{P}$); hydrodynamic interactions for spheres (Rotne-Prager-Yamakawa tensor); corrections for double layer kernels (e.g., for stresslets, rotlet, and Stokes doublet in Section 2), etc. The Laplace routines of StokesFMMLib3D library compute sums of the form:

$$\phi(\mathbf{y}^m) \;\; = \;\; \sum_{n=1}^{N} q^n\, G(\mathbf{y}^m - \mathbf{x}^n) + p^n\, \boldsymbol{\nu}^n \cdot \nabla_{x^n} G(\mathbf{y}^m - \mathbf{x}^n) + h^n\, \boldsymbol{\eta}^n \cdot \nabla_{x^n}\nabla_{x^n} G(\mathbf{y}^m - \mathbf{x}^n),$$

for $m = 1, \ldots, M$, where $G(\mathbf{x}) = 1/||\mathbf{x}||$, q^n are referred to as the charge strengths, p^n as the dipole strengths, h^n as the quadrupole strengths, and $\boldsymbol{\nu}^n$, $\boldsymbol{\eta}^n$ are the vectors whose directions determine the dipole and quadrupole orientations, respectively (if present). We use a non-standard way to represent the quadrupole orientation tensor, and take advantage of the symmetry to express it as a six-dimensional vector. More precisely, the sums computed are of the form

$$\begin{aligned}\phi(\mathbf{y}^m) &= \sum_{n=1}^{N} q^n\, G(\mathbf{y}^m - \mathbf{x}^n) + p^n\, (\nu_1^n \cdot \partial_{x_1^n} + \nu_2^n \cdot \partial_{x_2^n} + \nu_3^n \cdot \partial_{x_3^n}) G(\mathbf{y}^m - \mathbf{x}^n) \\ &\quad + h^n\, (\eta_1^n \cdot \partial^2_{x_{11}^n} + \eta_2^n \cdot \partial^2_{x_{22}^n} + \eta_3^n \cdot \partial^2_{x_{33}^n} + \eta_4^n \cdot \partial^2_{x_{12}^n} + \eta_5^n \cdot \partial^2_{x_{13}^n} + \eta_6^n \cdot \partial^2_{x_{23}^n}) G(\mathbf{y}^m - \mathbf{x}^n).\end{aligned}$$

Important note: The charge and dipole strengths are assumed to be **complex** double precision numbers for the Laplace library. 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 <u>not</u> the most highly optimized version possible, intended rather to be accessible and modifiable with only modest effort. The translation operators used in StokesLib3D 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.

In the next sections, we describe the calling sequences for the Fortran routines. The corresponding MATLAB routines are described in Contents.m in the matlab subdirectory.

4 StokesFMMLib3D: Free space Green's functions

4.1 Subroutine STFMM3DPARTSELF

---

subroutine stfmm3dpartself (ier,iprec,nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$u_i(\mathbf{x}^m)/4\pi = \sum_{\substack{n=1\\n\neq m}}^{N} \sum_{j=1}^{3} S_{ij}(\mathbf{x}^m - \mathbf{x}^n)\, f_j^m + \sum_{\substack{n=1\\n\neq m}}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} T_{ijk}(\mathbf{x}^m - \mathbf{x}^n)\, \nu_k^n\, g_j^n, \tag{11}$$

$$p(\mathbf{x}^m)/4\pi = \sum_{\substack{n=1\\n\neq m}}^{N} \sum_{j=1}^{3} P_j(\mathbf{x}^m - \mathbf{x}^n)\, f_j^n + \sum_{\substack{n=1\\n\neq m}}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} \Pi_{jk}(\mathbf{x}^m - \mathbf{x}^n)\, \nu_k^n\, g_j^n, \tag{12}$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$.

Input Parameters:

`iprec` ***integer*** :

precision flag. Allowed values are

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

`nparts` ***integer*** :

number of sources

`source(3,nparts)` ***real *8*** :

sources(k,j) is the kth component of the jth source $\mathbf{x}^j$ in $\mathbf{R}^3$.

`ifsingle` ***integer*** :

single force flag. If `ifsingle` $= 0$, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :

sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :

double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force sources. Allowed valued are

`ifdouble` $= 1$, standard stresslet (type 1),
$$T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right)$$

`ifdouble` $= 2$, symmetric stresslet (type 2),
$$T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

`ifdouble` $= 3$, rotlet,
$$T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0,$$

`ifdouble` $= 4$, Stokes doublet,
$$T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :

sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :

sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :

velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :

velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

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(3,nparts)` ***real *8*** :

$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :

$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :

$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

4.2 Subroutine STFMM3DPARTTARG

subroutine stfmm3dparttarg (ier,iprec,nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad, ntargs,target,ifpottarg,pottarg,pretarg,ifgradtarg,gradtarg)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$
\begin{aligned}
u_i(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} S_{ij}(\mathbf{x}^m-\mathbf{x}^n)\, f_j^m + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T_{ijk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu_k^n\, g_j^n,\\
p(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} P_j(\mathbf{x}^m-\mathbf{x}^n)\, f_j^n + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi_{jk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu_k^n\, g_j^n,
\end{aligned}
$$

$$
\begin{aligned}
u_i(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} S_{ij}(\mathbf{y}^m-\mathbf{x}^n)\, f_j^m + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T_{ijk}(\mathbf{y}^m-\mathbf{x}^n)\,\nu_k^n\, g_j^n,\\
p(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} P_j(\mathbf{y}^m-\mathbf{x}^n)\, f_j^n + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi_{jk}(\mathbf{y}^m-\mathbf{x}^n)\,\nu_k^n\, g_j^n,
\end{aligned}
$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$.

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 $< 0.5\,10^{-14}$.

`nparts` ***integer*** :
number of sources

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

`ifsingle` ***integer*** :
single force flag. If `ifsingle` $= 0$, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :
sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :
double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force sources. Allowed valued are

`ifdouble` $= 1$, standard stresslet (type 1),
$$T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right)$$
`ifdouble` $= 2$, symmetric stresslet (type 2),
$$T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$
`ifdouble` $= 3$, rotlet,
$$T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0,$$
`ifdouble` $= 4$, Stokes doublet,
$$T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :
sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :
sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :
velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :
velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

`ntargs` ***integer*** :
number of targets

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

`ifpottarg` ***integer*** :
target velocity field/pressure flag. If `ifpottarg` $= 1$, the target velocity field and pressure are computed. Otherwise, they are not.

`ifgradtarg` ***integer*** :
target velocity gradient flag. If `ifgradtarg` $= 1$ the gradient of the target velocity field is computed. Otherwise, it is not.

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(3,nparts)` ***real *8*** :

$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :

$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :

$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

`pottarg(3,ntargs)` ***real *8*** :

$u_i(\mathbf{y}^m)$ - pottarg(i,m) is the ith component of the velocity field at the mth target.

`pretarg(ntargs)` ***real *8*** :

$p(\mathbf{y}^m)$ - pretarg(m) is the pressure at the mth target.

`gradtarg(3,3,ntargs)` ***real *8*** :

$\partial_k u_i(\mathbf{y}^m)$ - gradtarg(i,k,m) is the k-th derivative of ith component of the velocity field at the mth target.

4.3 Subroutine ST3DPARTDIRECT

---

subroutine st3dpartdirect (nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad, ntargs,target,ifpottarg,pottarg,pretarg,ifgradtarg,gradtarg)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$
\begin{aligned}
u_i(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} S_{ij}(\mathbf{x}^m-\mathbf{x}^n)\, f_j^m + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T_{ijk}(\mathbf{x}^m-\mathbf{x}^n)\, \nu_k^n\, g_j^n, \\
p(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} P_j(\mathbf{x}^m-\mathbf{x}^n)\, f_j^n + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi_{jk}(\mathbf{x}^m-\mathbf{x}^n)\, \nu_k^n\, g_j^n,
\end{aligned}
$$

$$
\begin{aligned}
u_i(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} S_{ij}(\mathbf{y}^m-\mathbf{x}^n)\, f_j^m + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T_{ijk}(\mathbf{y}^m-\mathbf{x}^n)\, \nu_k^n\, g_j^n, \\
p(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} P_j(\mathbf{y}^m-\mathbf{x}^n)\, f_j^n + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi_{jk}(\mathbf{y}^m-\mathbf{x}^n)\, \nu_k^n\, g_j^n,
\end{aligned}
$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$. **It implements the summation formula directly and is not fast.**

Input Parameters:

`nparts` ***integer*** :
  number of sources

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

`ifsingle` ***integer*** :
  single force flag. If `ifsingle` $= 0$, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :
  sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :
  double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force

sources. Allowed valued are

`ifdouble` $= 1$, standard stresslet (type 1),

$$T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right)$$

`ifdouble` $= 2$, symmetric stresslet (type 2),

$$T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

`ifdouble` $= 3$, rotlet,

$$T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0,$$

`ifdouble` $= 4$, Stokes doublet,

$$T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :
sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :
sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :
velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :
velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

`ntargs` ***integer*** :
number of targets

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

`ifpottarg` ***integer*** :
target velocity field/pressure flag. If `ifpottarg` $= 1$, the target velocity field and pressure are computed. Otherwise, they are not.

`ifgradtarg` ***integer*** :
target velocity gradient flag. If `ifgradtarg` $= 1$ the gradient of the target velocity field is computed. Otherwise, it is not.

Output Parameters:

`pot(3,nparts)` ***real *8*** :
$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :
$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :
$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

`pottarg(3,ntargs)` ***real *8*** :
$u_i(\mathbf{y}^m)$ - pottarg(i,m) is the ith component of the velocity field at the mth target.

`pretarg(ntargs)` ***real *8*** :
$p(\mathbf{y}^m)$ - pretarg(m) is the pressure at the mth target.

`gradtarg(3,3,ntargs)` ***real *8*** :
$\partial_k u_i(\mathbf{y}^m)$ - gradtarg(i,k,m) is the k-th derivative of ith component of the velocity field at the mth target.

4.4 Subroutine LFMM3DPARTQUADSELF

---

subroutine lfmm3dpartquadself(ier, iprec, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifquad, quadstr, quadvec, ifpot, pot, iffld, fld, ifhess, hess)

computes sums of the form

$$\phi(\mathbf{x}^i) = \sum_{\substack{j=1 \\ i \neq j}}^{N} q^j \frac{1}{\|\mathbf{x}^i - \mathbf{x}^j\|} + p^j \boldsymbol{\nu}^j \cdot \nabla_{\mathbf{x}^j} \left(\frac{1}{\|\mathbf{x}^i - \mathbf{x}^j\|} \right) + h^j \boldsymbol{\eta}^j \cdot \nabla_{\mathbf{x}^j} \nabla_{\mathbf{x}^j} \left(\frac{1}{\|\mathbf{x}^i - \mathbf{x}^j\|} \right) \quad (13)$$

for $i = 1, \ldots, N$, as well as fields (-gradients) and hessians (second derivatives) of ϕ.

Input Parameters:

`iprec` ***integer*** :

precision flag. Allowed values are

$$\begin{array}{ll} \texttt{iprec} = -2 & \text{for least squares errors} < 0.5\,10^{0}, \\ \texttt{iprec} = -1 & \text{for least squares errors} < 0.5\,10^{-1}, \\ \texttt{iprec} = 0 & \text{for least squares errors} < 0.5\,10^{-2}, \\ \texttt{iprec} = 1 & \text{for least squares errors} < 0.5\,10^{-3}. \\ \texttt{iprec} = 2 & \text{for least squares errors} < 0.5\,10^{-6}. \\ \texttt{iprec} = 3 & \text{for least squares errors} < 0.5\,10^{-9}. \\ \texttt{iprec} = 4 & \text{for least squares errors} < 0.5\,10^{-12}. \\ \texttt{iprec} = 5 & \text{for least squares errors} < 0.5\,10^{-14}. \end{array}$$

`nsource` ***integer*** :

number of sources

`source(3,nsources)` ***real *8*** :

sources(k,j) is the kth component of the jth source $\mathbf{x}^j$ in $\mathbf{R}^3$.

`ifcharge` ***integer*** :

charge flag. If `ifcharge` = 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 (13)).

`ifdipole` ***integer*** :

dipole flag. If `ifdipole` = 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 (13)).

`dipvec(3,nsources)` ***real *8*** :
: dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\boldsymbol{\nu}^j$ in the formula (13)).

`ifquad` ***integer*** :
: quadrupole flag. If `ifquad` = 1, then include the effect of the quadrupole sources. Otherwise, omit.

`quadstr(nsources)` ***complex *16*** :
: quadstr(j) is the strength of the jth quadrupole (h_j in the formula (13)).

`quadvec(6,nsources)` ***real *8*** :
: quadvec(k,j) is the kth component of the orientation vector of the jth quadrupole ($\boldsymbol{\eta}_j$ in the formula (13)). quadvec(1,i), quadvec(2,i), quadvec(3,i), quadvec(4,i), quadvec(5,i), and quadvec(6,i) correspond to (11), (22), (33), (12), (13), and (23) components of the quadrupole orientation tensor, respectively.

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

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

`ifhess` ***integer*** :
: hessian flag. If `ifhess` = 1 the hessian (second derivatives) 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). ifquad = 0, quadstr can be dimensioned as a complex scalar and quadvec can be dimensioned in the calling program as quadvec(6) - BUT NOT quadvec(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

`hess(6,nsources)` ***complex *16*** :

hess(k,i) is the kth component of the hessian (second derivatives) of the potential) at the ith source. hess(1,i), hess(2,i), hess(3,i), hess(4,i), hess(5,i), and hess(6,i) are ordered as $(\partial_{x_1x_1}, \partial_{x_2x_2}, \partial_{x_3x_3}, \partial_{x_1x_2}, \partial_{x_1x_3}, \partial_{x_2x_3})$.

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

4.5 Subroutine LFMM3DPARTQUADTARG

---

subroutine lfmm3dpartquadtarg(ier, iprec, nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifquad, quadstr, quadvec, ifpot, pot, iffld, fld, ifhess, hess, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg, ifhesstarg, hesstarg)

computes sums of the form

$$\begin{aligned}
\phi(\mathbf{x}^i) &= \sum_{\substack{j=1\\ i\neq j}}^{N} q^j \frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|} + p^j \boldsymbol{\nu}^j \cdot \nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|}\right) + h^j \boldsymbol{\eta}^j \cdot \nabla_{\mathbf{x}^j}\nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|}\right)\\
\phi(\mathbf{y}^i) &= \sum_{j=1}^{N} q^j \frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|} + p^j \boldsymbol{\nu}^j \cdot \nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|}\right) + h^j \boldsymbol{\eta}^j \cdot \nabla_{\mathbf{x}^j}\nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|}\right)
\end{aligned}$$

for $i = 1, \ldots, N$, as well as fields (-gradients) and hessians (second derivatives) of ϕ.

Input Parameters:

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

$$\begin{array}{ll}
\texttt{iprec} = -2 & \text{for least squares errors} < 0.5\,10^{0},\\
\texttt{iprec} = -1 & \text{for least squares errors} < 0.5\,10^{-1},\\
\texttt{iprec} = 0 & \text{for least squares errors} < 0.5\,10^{-2},\\
\texttt{iprec} = 1 & \text{for least squares errors} < 0.5\,10^{-3}.\\
\texttt{iprec} = 2 & \text{for least squares errors} < 0.5\,10^{-6}.\\
\texttt{iprec} = 3 & \text{for least squares errors} < 0.5\,10^{-9}.\\
\texttt{iprec} = 4 & \text{for least squares errors} < 0.5\,10^{-12}.\\
\texttt{iprec} = 5 & \text{for least squares errors} < 0.5\,10^{-14}.
\end{array}$$

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

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

`ifcharge` ***integer*** :
charge flag. If `ifcharge` $= 1$, 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 (13)).

`ifdipole` ***integer*** :
dipole flag. If `ifdipole` $= 1$, 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 (13)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\boldsymbol{\nu}^j$ in the formula (13)).

`ifquad` ***integer*** :
quadrupole flag. If `ifquad` = 1, then include the effect of the quadrupole sources. Otherwise, omit.

`quadstr(nsources)` ***complex *16*** :
quadstr(j) is the strength of the jth quadrupole (h_j in the formula (13)).

`quadvec(6,nsources)` ***real *8*** :
quadvec(k,j) is the kth component of the orientation vector of the jth quadrupole ($\boldsymbol{\eta}_j$ in the formula (13)). quadvec(1,i), quadvec(2,i), quadvec(3,i), quadvec(4,i), quadvec(5,i), and quadvec(6,i) correspond to (11), (22), (33), (12), (13), and (23) components of the quadrupole orientation tensor, respectively.

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

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

`ifhess` ***integer*** :
hessian flag. If `ifhess` = 1 the hessian (second derivatives) of the potential is computed. Otherwise, it is not.

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

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

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

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

`ifhesstarg` ***integer*** :
target hessian flag. If `ifhess` = 1 the hessian (second derivatives) 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). ifquad = 0, quadstr can be dimensioned as a complex scalar and quadvec can be dimensioned in the calling program as quadvec(6) - BUT NOT quadvec(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

`hess(6,nsources)` ***complex *16*** :

hess(k,i) is the kth component of the hessian (second derivatives) of the potential) at the ith source. hess(1,i), hess(2,i), hess(3,i), hess(4,i), hess(5,i), and hess(6,i) are ordered as $(\partial_{x_1x_1}, \partial_{x_2x_2}, \partial_{x_3x_3}, \partial_{x_1x_2}, \partial_{x_1x_3}, \partial_{x_2x_3})$.

`pottarg(ntargets)` ***complex *16*** :

pottarg(i) is the potential at the ith target

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

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

`hesstarg(6,ntargets)` ***complex *16*** :

hesstarg(k,i) is the kth component of the hessian (second derivatives) of the potential) at the ith target. hesstarg(1,i), hesstarg(2,i), hesstarg(3,i), hesstarg(4,i), hesstarg(5,i), and hesstarg(6,i) are ordered as $(\partial_{x_1x_1}, \partial_{x_2x_2}, \partial_{x_3x_3}, \partial_{x_1x_2}, \partial_{x_1x_3}, \partial_{x_2x_3})$.

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

4.6 Subroutine L3DPARTQUADDIRECT

---

subroutine l3dpartquaddirect(nsource, source, ifcharge, charge, ifdipole, dipstr, dipvec, ifquad, quadstr, quadvec, ifpot, pot, iffld, fld, ifhess, hess, ntarget, target, ifpottarg, pottarg, iffldtarg, fldtarg, ifhesstarg, hesstarg)

computes sums of the form

$$\begin{aligned}
\phi(\mathbf{x}^i) &= \sum_{\substack{j=1\\ i\neq j}}^{N} q^j \frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|} + p^j \boldsymbol{\nu}^j \cdot \nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|}\right) + h^j \boldsymbol{\eta}^j \cdot \nabla_{\mathbf{x}^j}\nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{x}^i-\mathbf{x}^j\|}\right)\\
\phi(\mathbf{y}^i) &= \sum_{j=1}^{N} q^j \frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|} + p^j \boldsymbol{\nu}^j \cdot \nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|}\right) + h^j \boldsymbol{\eta}^j \cdot \nabla_{\mathbf{x}^j}\nabla_{\mathbf{x}^j}\left(\frac{1}{\|\mathbf{y}^i-\mathbf{x}^j\|}\right)
\end{aligned}$$

for $i = 1, \ldots, N$, as well as fields (-gradients) and hessians (second derivatives) of ϕ. **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 $\mathbf{x}^j$ in $\mathbf{R}^3$.

`ifcharge` ***integer*** :
charge flag. If `ifcharge` = 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 (13)).

`ifdipole` ***integer*** :
dipole flag. If `ifdipole` = 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 (13)).

`dipvec(3,nsources)` ***real *8*** :
dipvec(k,j) is the kth component of the orientation vector of the jth dipole ($\boldsymbol{\nu}^j$ in the formula (13)).

`ifquad` ***integer*** :
quadrupole flag. If `ifquad` = 1, then include the effect of the quadrupole sources. Otherwise, omit.

`quadstr(nsources)` ***complex *16*** :
quadstr(j) is the strength of the jth quadrupole (h_j in the formula (13)).

`quadvec(6,nsources)` ***real *8*** :
quadvec(k,j) is the kth component of the orientation vector of the jth quadrupole ($\boldsymbol{\eta}_j$ in the formula (13)). quadvec(1,i), quadvec(2,i), quadvec(3,i), quadvec(4,i), quadvec(5,i), and quadvec(6,i) correspond to (11), (22), (33), (12), (13), and (23) components of the quadrupole orientation tensor, respectively.

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

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

`ifhess` ***integer*** :
hessian flag. If `ifhess` = 1 the hessian (second derivatives) of the potential is computed. Otherwise, it is not.

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

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

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

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

`ifhesstarg` ***integer*** :
target hessian flag. If `ifhess` = 1 the hessian (second derivatives) 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). ifquad = 0, quadstr can be dimensioned as a complex scalar and quadvec can be dimensioned in the calling program as quadvec(6) - BUT NOT quadvec(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

`hess(6,nsources)` ***complex *16*** :
hess(k,i) is the kth component of the hessian (second derivatives) of the potential) at the ith source. hess(1,i), hess(2,i), hess(3,i), hess(4,i), hess(5,i), and hess(6,i) are ordered as $(\partial_{x_1x_1}, \partial_{x_2x_2}, \partial_{x_3x_3}, \partial_{x_1x_2}, \partial_{x_1x_3}, \partial_{x_2x_3})$.

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

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

`hesstarg(6,ntargets)` ***complex *16*** :
hesstarg(k,i) is the kth component of the hessian (second derivatives) of the potential) at the ith target. hesstarg(1,i), hesstarg(2,i), hesstarg(3,i), hesstarg(4,i), hesstarg(5,i), and hesstarg(6,i) are ordered as $(\partial_{x_1x_1}, \partial_{x_2x_2}, \partial_{x_3x_3}, \partial_{x_1x_2}, \partial_{x_1x_3}, \partial_{x_2x_3})$.

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

---

5 StokesFMMLib3D: Half space Green's functions

5.1 Subroutine STHFMM3DPARTSELF

---

subroutine sthfmm3dpartself (ier,iprec,itype,nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$u_i(\mathbf{x}^m)/4\pi = \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} S^h_{ij}(\mathbf{x}^m-\mathbf{x}^n)\, f^m_j + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T^h_{ijk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j, \tag{14}$$

$$p(\mathbf{x}^m)/4\pi = \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} P^h_{j}(\mathbf{x}^m-\mathbf{x}^n)\, f^n_j + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi^h_{jk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j, \tag{15}$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$. A half space no-slip boundary condition is assumed, i.e. $\mathbf{u} = 0$ at $z = 0$.

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 $< 0.5\,10^{-14}$.

`itype` ***integer*** :
half space evaluation flag. If `itype` $= 1$, then include the effects of both direct arrival and the image contribution. If `itype` $= 2$, then include the effects of the image contribution only.

`nparts` ***integer*** :
number of sources

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

`ifsingle` ***integer*** :
single force flag. If `ifsingle` $= 0$, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :
sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :
double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force sources. Allowed valued are

$$
\begin{array}{ll}
\texttt{ifdouble} = 1, & \text{standard stresslet (type 1)}, \\
 & T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right) \\
\texttt{ifdouble} = 2, & \text{symmetric stresslet (type 2)}, \\
 & T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right), \\
\texttt{ifdouble} = 3, & \text{rotlet}, \\
 & T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0, \\
\texttt{ifdouble} = 4, & \text{Stokes doublet}, \\
 & T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),
\end{array}
$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :
sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :
sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :
velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :
velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

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(3,nparts)` ***real *8*** :
$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :
$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :

$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

5.2 Subroutine STHFMM3DPARTTARG

---

subroutine sthfmm3dparttarg (ier,iprec,itype,nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad, ntargs,target,ifpottarg,pottarg,pretarg,ifgradtarg,gradtarg)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$
\begin{aligned}
u_i(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\ n\neq m}}^{N} \sum_{j=1}^{3} S^h_{ij}(\mathbf{x}^m - \mathbf{x}^n)\, f^m_j + \sum_{\substack{n=1\\ n\neq m}}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} T^h_{ijk}(\mathbf{x}^m - \mathbf{x}^n)\, \nu^n_k\, g^n_j, \\
p(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\ n\neq m}}^{N} \sum_{j=1}^{3} P^h_{j}(\mathbf{x}^m - \mathbf{x}^n)\, f^n_j + \sum_{\substack{n=1\\ n\neq m}}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} \Pi^h_{jk}(\mathbf{x}^m - \mathbf{x}^n)\, \nu^n_k\, g^n_j,
\end{aligned}
$$

$$
\begin{aligned}
u_i(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N} \sum_{j=1}^{3} S^h_{ij}(\mathbf{y}^m - \mathbf{x}^n)\, f^m_j + \sum_{n=1}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} T^h_{ijk}(\mathbf{y}^m - \mathbf{x}^n)\, \nu^n_k\, g^n_j, \\
p(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N} \sum_{j=1}^{3} P^h_{j}(\mathbf{y}^m - \mathbf{x}^n)\, f^n_j + \sum_{n=1}^{N} \sum_{j=1}^{3} \sum_{k=1}^{3} \Pi^h_{jk}(\mathbf{y}^m - \mathbf{x}^n)\, \nu^n_k\, g^n_j,
\end{aligned}
$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$. A half space no-slip boundary condition is assumed, i.e. $\mathbf{u} = 0$ at $z = 0$.

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 $< 0.5\,10^{-14}$.

`itype` ***integer*** :
half space evaluation flag. If `itype` $= 1$, then include the effects of both direct arrival and the image contribution. If `itype` $= 2$, then include the effects of the image contribution only.

`nparts` ***integer*** :
number of sources

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

`ifsingle` ***integer*** :
single force flag. If `ifsingle` $= 0$, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :
sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :
double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force sources. Allowed valued are

`ifdouble` $= 1$, standard stresslet (type 1),
$$T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right)$$
`ifdouble` $= 2$, symmetric stresslet (type 2),
$$T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$
`ifdouble` $= 3$, rotlet,
$$T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0,$$
`ifdouble` $= 4$, Stokes doublet,
$$T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :
sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :
sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :
velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :
velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

`ntargs` ***integer*** :
number of targets

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

`ifpottarg` ***integer*** :
target velocity field/pressure flag. If `ifpottarg` $= 1$, the target velocity field and pressure are computed. Otherwise, they are not.

`ifgradtarg` ***integer*** :
target velocity gradient flag. If `ifgradtarg` $= 1$ the gradient of the target velocity field is computed. Otherwise, it is not.

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(3,nparts)` ***real *8*** :
$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :
$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :
$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

`pottarg(3,ntargs)` ***real *8*** :
$u_i(\mathbf{y}^m)$ - pottarg(i,m) is the ith component of the velocity field at the mth target.

`pretarg(ntargs)` ***real *8*** :
$p(\mathbf{y}^m)$ - pretarg(m) is the pressure at the mth target.

`gradtarg(3,3,ntargs)` ***real *8*** :
$\partial_k u_i(\mathbf{y}^m)$ - gradtarg(i,k,m) is the k-th derivative of ith component of the velocity field at the mth target.

5.3 Subroutine STH3DPARTDIRECT

---

subroutine sth3dpartdirect (itype,nparts,source, ifsingle,sigma_sl,ifdouble,sigma_dl,sigma_dv, ifpot,pot,pre,ifgrad,grad, ntargs,target,ifpottarg,pottarg,pretarg,ifgradtarg,gradtarg)

computes sums of the form (omitting the normalization factor $\frac{1}{4\pi}$):

$$
\begin{aligned}
u_i(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} S^h_{ij}(\mathbf{x}^m-\mathbf{x}^n)\, f^m_j + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T^h_{ijk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j,\\
p(\mathbf{x}^m)/4\pi &= \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3} P^h_{j}(\mathbf{x}^m-\mathbf{x}^n)\, f^n_j + \sum_{\substack{n=1\\n\neq m}}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi^h_{jk}(\mathbf{x}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j,
\end{aligned}
$$

$$
\begin{aligned}
u_i(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} S^h_{ij}(\mathbf{y}^m-\mathbf{x}^n)\, f^m_j + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} T^h_{ijk}(\mathbf{y}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j,\\
p(\mathbf{y}^m)/4\pi &= \sum_{n=1}^{N}\sum_{j=1}^{3} P^h_{j}(\mathbf{y}^m-\mathbf{x}^n)\, f^n_j + \sum_{n=1}^{N}\sum_{j=1}^{3}\sum_{k=1}^{3} \Pi^h_{jk}(\mathbf{y}^m-\mathbf{x}^n)\,\nu^n_k\, g^n_j,
\end{aligned}
$$

for $i = 1, \ldots, N$, as well as gradients of $\mathbf{u}$. A half space no-slip boundary condition is assumed, i.e. $\mathbf{u} = 0$ at $z = 0$. **It implements the summation formula directly and is not fast.**

Input Parameters:

`itype` ***integer*** :
: half space evaluation flag. If `itype` = 1, then include the effects of both direct arrival and the image contribution. If `itype` = 2, then include the effects of the image contribution only.

`nparts` ***integer*** :
: number of sources

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

`ifsingle` ***integer*** :
: single force flag. If `ifsingle` = 0, then exclude the effect of the single force sources. Otherwise, omit.

`sigma_sl(3,nparts)` ***real *8*** :
: sigma_sl(3,n) is the strength of the nth single force source (f^n in the formula (12)).

`ifdouble` ***integer*** :
double force flag. If `ifdouble` $= 1, 2, 3, 4$, then include the effect of the double force sources. Allowed valued are

$$\begin{array}{ll}
\texttt{ifdouble} = 1, & \text{standard stresslet (type 1)}, \\
& T_{ijk}(\mathbf{x}) = \frac{6}{8\pi}\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}, \quad \Pi_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right) \\
\texttt{ifdouble} = 2, & \text{symmetric stresslet (type 2)}, \\
& T^{(2)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5}\right), \quad \Pi^{(2)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right), \\
\texttt{ifdouble} = 3, & \text{rotlet}, \\
& T^{(3)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(\frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(3)}_{jk} = 0, \\
\texttt{ifdouble} = 4, & \text{Stokes doublet}, \\
& T^{(4)}_{ijk}(\mathbf{x}) = \frac{2}{8\pi}\left(-\frac{\mathbf{x}_i\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_j\mathbf{x}_k}{||\mathbf{x}||^5} + \frac{\mathbf{x}_k\delta_{ij}}{||\mathbf{x}||^3} - \frac{\mathbf{x}_j\delta_{ik}}{||\mathbf{x}||^3}\right), \quad \Pi^{(4)}_{jk} = \frac{4}{8\pi}\left(-\frac{\delta_{jk}}{||\mathbf{x}||^3} + 3\frac{\mathbf{x}_i\mathbf{x}_k}{||\mathbf{x}||^5}\right),
\end{array}$$

otherwise, omit.

`sigma_dl(3,nparts)` ***real *8*** :
sigma_dl(3,n) is the strength of the nth double force source (g^n in the formula (12)).

`sigma_dv(3,nparts)` ***real *8*** :
sigma_dv(3,n) is the orientation of the nth double force source (ν^n in the formula (12)).

`ifpot` ***integer*** :
velocity field/pressure flag. If `ifpot` $= 1$, the velocity field and pressure are computed. Otherwise, they are not.

`ifgrad` ***integer*** :
velocity gradient flag. If `ifgrad` $= 1$ the gradient of the velocity field is computed. Otherwise, it is not.

`ntargs` ***integer*** :
number of targets

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

`ifpottarg` ***integer*** :
target velocity field/pressure flag. If `ifpottarg` $= 1$, the target velocity field and pressure are computed. Otherwise, they are not.

`ifgradtarg` ***integer*** :
target velocity gradient flag. If `ifgradtarg` $= 1$ the gradient of the target velocity field is computed. Otherwise, it is not.

Output Parameters:

`pot(3,nparts)` ***real *8*** :
$u_i(\mathbf{x}^m)$ - pot(i,m) is the ith component of the velocity field at the mth source.

`pre(nparts)` ***real *8*** :
$p(\mathbf{x}^m)$ - pre(m) is the pressure at the mth source.

`grad(3,3,nparts)` ***real *8*** :
$\partial_k u_i(\mathbf{x}^m)$ - grad(i,k,m) is the k-th derivative of ith component of the velocity field at the mth source.

`pottarg(3,ntargs)` ***real *8*** :
$u_i(\mathbf{y}^m)$ - pottarg(i,m) is the ith component of the velocity field at the mth target.

`pretarg(ntargs)` ***real *8*** :
$p(\mathbf{y}^m)$ - pretarg(m) is the pressure at the mth target.

`gradtarg(3,3,ntargs)` ***real *8*** :
$\partial_k u_i(\mathbf{y}^m)$ - gradtarg(i,k,m) is the k-th derivative of ith component of the velocity field at the mth target.

6 Sample drivers for StokesFMMLib3D

In the STFMM3D/examples directory, the file `stfmm3dpart_dr.f` contains a sample driver for `stfmm3dparttarg`. 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 $(1, 0, -2)$. The code then computes the velocity field, pressure and velocity gradient 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 four or five seconds.

In the STFMM3D/examples directory, the file `lfmm3dpartquad_dr.f` contains a sample driver for `lfmm3dpartquadtarg`. 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 $(1, 0, 0)$. 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.

7 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.