WBFMM

Generated by Doxygen 1.8.9.1

Tue Aug 20 2019 09:15:53

Contents

Chapter 1

WBFMM: A Wide Band Fast Multipole Method library

WBFMM is a library and collection of associated tools for the efficient solution of the Helmholtz equation and in particular the summation of fields generated by large numbers of acoustic sources.

1.1 Getting started

1.2 What WBFMM does

1.3 References

The following papers and links have been used in some way in developing WBFMM:

1. Nail A. Gumerov and Ramani Duraiswami, Recursions for the Computation of Multipole Translation and Rotation Coefficients for the 3-D Helmholtz Equation, SIAM J. Sci. Comput., 25(4), 1344-1381, `http://dx.doi.org/10.1137/S1064827501399705`

2. Gumerov, Duraiswami, and Borovikov, Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in d Dimensions, 2003, `http://users.umiacs.umd.edu/~gumerov/PDFs/cs-tr-4458.pdf`

3. Nail A. Gumerov and Ramani Duraiswami, A broadband fast multipole accelerated boundary element method for the three dimensional Helmholtz equation, J. Acoust. Soc. Am., 125(1), `http://dx.doi.org/10.1121/1.3021297`

4. Nail A. Gumerov and Ramani Duraiswami, Comparison of the efficiency of translation operators used in the fast multipole method for the 3D Laplace equation, 2005, `http://www.umiacs.umd.edu/~ramani/pubs/comparisontranslationmethods_041205.pdf`

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 4

File Index

4.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 5

Module Documentation

5.1 Boxes and octrees

Operations on octree boxes and trees.

Data Structures

- struct **wbfmm_box_t**
- struct **wbfmm_tree_t**
- struct **wbfmm_shift_operators_t**

Enumerations

- enum **wbfmm_problem_t** { **WBFMM_PROBLEM_LAPLACE** = 1, **WBFMM_PROBLEM_HELMHOLTZ** = 2 }

Functions

- gint **wbfmm_tree_add_level** (**wbfmm_tree_t** ∗t)
- gint **wbfmm_tree_add_points** (**wbfmm_tree_t** ∗t, gpointer pts, guint npts, gsize stride)

  Add points to an octree.
- **wbfmm_tree_t** ∗ **wbfmm_tree_new** (gdouble ∗x, gdouble D, guint maxpoints)

  Allocate a new octree.
- gint **wbfmm_tree_leaf_expansions** (**wbfmm_tree_t** ∗t, gdouble k, gdouble ∗src, gint sstr, gdouble ∗normals, gint nstr, gdouble ∗dipoles, gint dstr, gboolean zero_expansions, gdouble ∗work)

  Generate leaf expansions for a tree.
- guint64 **wbfmm_point_index_3d** (gdouble ∗x, gdouble ∗c, gdouble D)

  Find Morton index for point in a cubic domain.
- gint **wbfmm_tree_coefficient_init** (**wbfmm_tree_t** ∗t, guint l, guint nr, guint ns)

  Initialize expansion coefficient data in an octree.
- gint **wbfmm_tree_refine** (**wbfmm_tree_t** ∗t)

  Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.
- gint **wbfmm_tree_add_points_f** (**wbfmm_tree_t** ∗t, gpointer pts, guint npts, gsize stride)

  Add points to an octree.
- **wbfmm_tree_t** ∗ **wbfmm_tree_new_f** (gfloat ∗x, gfloat D, guint maxpoints)

  Allocate a new octree.

- gint **wbfmm_tree_leaf_expansions_f** (**wbfmm_tree_t** ∗t, gfloat k, gfloat ∗src, gint sstr, gfloat ∗normals, gint nstr, gfloat ∗dipoles, gint dstr, gboolean zero_expansions, gfloat ∗work)

  Generate leaf expansions for a tree.
- guint64 **wbfmm_point_index_3d_f** (gfloat ∗x, gfloat ∗c, gfloat D)

  Find Morton index for point in a cubic domain.
- gint **wbfmm_tree_coefficient_init_f** (**wbfmm_tree_t** ∗t, guint l, guint nr, guint ns)

  Initialize expansion coefficient data in an octree.
- gint **wbfmm_tree_refine_f** (**wbfmm_tree_t** ∗t)

  Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.

5.1.1 Detailed Description

Operations on octree boxes and trees.

5.1.2 Enumeration Type Documentation

5.1.2.1 enum wbfmm_problem_t

Selection of physical problem to be handled by a **wbfmm_tree_t** (p. 48)

Enumerator

WBFMM_PROBLEM_LAPLACE Laplace equation

WBFMM_PROBLEM_HELMHOLTZ Helmholtz equation

5.1.3 Function Documentation

5.1.3.1 guint64 wbfmm_point_index_3d (gdouble ∗ *x*, gdouble ∗ *c*, gdouble *D*)

Find Morton index for point in a cubic domain.

Parameters

x	point in space (three components, densely packed);
c	location of bottom left corner of domain;
D	width of domain.

Returns

0 on success

5.1.3.2 guint64 wbfmm_point_index_3d_f (gfloat ∗ *x*, gfloat ∗ *c*, gfloat *D*)

Find Morton index for point in a cubic domain.

Parameters

x	point in space (three components, densely packed);
c	location of bottom left corner of domain;
D	width of domain.

Returns

0 on success

5.1.3.3 gint wbfmm_tree_add_level (**wbfmm_tree_t** $*$ *t*)

Add a new level to an existing octree. The function assigns memory for, and initializes, a new layer of boxes of type **wbfmm_box_t** (p. 47)

Parameters

t	an existing **wbfmm_tree_t** (p. 48)

Returns

0 on success.

References wbfmm_tree_t::boxes, and wbfmm_tree_t::depth.

5.1.3.4 gint wbfmm_tree_add_points (wbfmm_tree_t * *t*, gpointer *pts*, guint *npts*, gsize *pstr*)

Add points to an octree.

Add a set of source points to an octree. The points are assumed to be in an array of real values with components in a packed triple, indexed using a stride of pstr bytes (this allows for quite general handling of different source formats).

Parameters

t	an existing **wbfmm_tree_t** (p. 48);
pts	an array containing point coordinates;
npts	the number of points in *pts*;
pstr	stride between points in bytes.

Returns

0 on success.

5.1.3.5 gint wbfmm_tree_add_points_f (wbfmm_tree_t * *t*, gpointer *pts*, guint *npts*, gsize *pstr*)

Add points to an octree.

Add a set of source points to an octree. The points are assumed to be in an array of real values with components in a packed triple, indexed using a stride of pstr bytes (this allows for quite general handling of different source formats).

Parameters

t	an existing **wbfmm_tree_t** (p. 48);
pts	an array containing point coordinates;
npts	the number of points in *pts*;
pstr	stride between points in bytes.

Returns

0 on success.

5.1.3.6 gint wbfmm_tree_coefficient_init (wbfmm_tree_t * *t*, guint *l*, guint *nr*, guint *ns*)

Initialize expansion coefficient data in an octree.

Parameters

t	octree for problem;
l	level to initialize data for;
nr	order of regular expansions at level *l*;
ns	order of singular expansions at level *l*.

Returns

0 on success

5.1.3.7 gint wbfmm_tree_coefficient_init_f (wbfmm_tree_t ∗ *t*, guint *l*, guint *nr*, guint *ns*)

Initialize expansion coefficient data in an octree.

Parameters

t	octree for problem;
l	level to initialize data for;
nr	order of regular expansions at level *l*;
ns	order of singular expansions at level *l*.

Returns

0 on success

5.1.3.8 gint wbfmm_tree_leaf_expansions (wbfmm_tree_t ∗ *t*, gdouble *k*, gdouble ∗ *src*, gint *sstr*, gdouble ∗ *normals*, gint *nstr*, gdouble ∗ *dipoles*, gint *dstr*, gboolean *zero_expansions*, gdouble ∗ *work*)

Generate leaf expansions for a tree.

Generate leaf expansions for a tree given some combination of monopole and dipole sources. Source positions are those in the point list attached to the tree using **wbfmm_tree_add_points** (p. 12)(...) and indexing in the array must correspond to that in the point list. Input arrays may be NULL: if *src* is not NULL, it is interpreted as a list of complex monopole strengths; if *normals* is not NULL, *dipoles* may not be NULL and they are interpreted respectively as a vector ('normal') at each source position and a scalar complex amplitude (this corresponds to surface normal and a normal velocity amplitude in a boundary element method calculation); if *normals* is NULL and *dipoles* is not NULL, *dipoles* is interpreted as a three-element complex vector specifying the dipole strength. The strides *sstr*, *nstr*, and *dstr* are the number of scalar elements between successive entries in the arrays, with the elements of each entry densely packed. For example, a list of normals might read:

$[n_{x1} \quad n_{y1} \quad n_{z1} \quad a_1 \quad b_1 \quad n_{x2} \ldots]$

where (n_{x1}, n_{y1}, n_{z1}) is the first normal vector and a_1 and b_1 are arbitrary entries in the array. In this case, the stride *nstr* would be 5, the number of elements between successive values of n_{xi}.

Parameters

t	octree for problem;
k	wavenumber;
src	monopole source strengths;
sstr	stride of data in *src*;
normals	dipole normals;
nstr	stride of data in *normals*;
dipoles	dipole source strengths (if *normals* is not NULL), or moment vectors (if *normals* is NULL);

dstr	stride of data in *dipoles*;
zero_expansions	if TRUE, set expansion coefficients to zero before adding source terms;
work	workspace.

Returns

0 on success

5.1.3.9 gint wbfmm_tree_leaf_expansions_f (wbfmm_tree_t ∗ *t*, gfloat *k*, gfloat ∗ *src*, gint *sstr*, gfloat ∗ *normals*, gint *nstr*, gfloat ∗ *dipoles*, gint *dstr*, gboolean *zero_expansions*, gfloat ∗ *work*)

Generate leaf expansions for a tree.

Generate leaf expansions for a tree given some combination of monopole and dipole sources. Source positions are those in the point list attached to the tree using **wbfmm_tree_add_points_f** (p. 12)(...) and indexing in the array must correspond to that in the point list. Input arrays may be NULL: if *src* is not NULL, it is interpreted as a list of complex monopole strengths; if *normals* is not NULL, *dipoles* may not be NULL and they are interpreted respectively as a vector ('normal') at each source position and a scalar complex amplitude (this corresponds to surface normal and a normal velocity amplitude in a boundary element method calculation); if *normals* is NULL and *dipoles* is not NULL, *dipoles* is interpreted as a three-element complex vector specifying the dipole strength. The strides *sstr*, *nstr*, and *dstr* are the number of scalar elements between successive entries in the arrays, with the elements of each entry densely packed. For example, a list of normals might read:

$[n_{x1} \quad n_{y1} \quad n_{z1} \quad a_1 \quad b_1 \quad n_{x2} \ldots]$

where (n_{x1}, n_{y1}, n_{z1}) is the first normal vector and a_1 and b_1 are arbitrary entries in the array. In this case, the stride *nstr* would be 5, the number of elements between successive values of n_{xi}.

Parameters

t	octree for problem;
k	wavenumber;
src	monopole source strengths;
sstr	stride of data in *src*;
normals	dipole normals;
nstr	stride of data in *normals*;
dipoles	dipole source strengths (if *normals* is not NULL), or moment vectors (if *normals* is NULL);
dstr	stride of data in *dipoles*;
zero_expansions	if TRUE, set expansion coefficients to zero before adding source terms;
work	workspace.

Returns

0 on success

5.1.3.10 wbfmm_tree_t ∗ wbfmm_tree_new (gdouble ∗ *x*, gdouble *D*, guint *maxpoints*)

Allocate a new octree.

Parameters

x	location of origin of tree;
D	width of domain;
maxpoints	maximum number of source points in tree.

Returns

pointer to newly allocated tree.

5.1.3.11 wbfmm_tree_t * wbfmm_tree_new_f (gfloat * *x*, gfloat *D*, guint *maxpoints*)

Allocate a new octree.

Parameters

x	location of origin of tree;
D	width of domain;
maxpoints	maximum number of source points in tree.

Returns

pointer to newly allocated tree.

5.1.3.12 gint wbfmm_tree_refine (wbfmm_tree_t * *t*)

Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.

Parameters

t	an existing **wbfmm_tree_t** (p. 48).

Returns

0 on success.

5.1.3.13 gint wbfmm_tree_refine_f (wbfmm_tree_t * *t*)

Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.

Parameters

t	an existing **wbfmm_tree_t** (p. 48).

Returns

0 on success.

5.2 Shift operations

Shift operations (combined rotation and translation) for upward and downward passes, and same-level interactions.

Functions

- gint **wbfmm_child_parent_shift** (gdouble ∗Cp, gint Np, gdouble ∗Cc, gint Nc, gdouble ∗H03, gdouble ∗H47, gint Lh, gdouble ∗shift, gint Ls, gdouble ∗work)

    Upward shift of singular expansion from eight children to common parent.
- gint **wbfmm_parent_child_shift** (gdouble ∗Cc, gint Nc, gdouble ∗Cp, gint Np, gdouble ∗H03, gdouble ∗H47, gint Lh, gdouble ∗shift, gint Ls, gdouble ∗work)

    Downward shift of parent expansion to child box centres.
- gint **wbfmm_shift_angles_list4** (gint i, gint j, gint k, gdouble ∗th, gdouble ∗ph, gdouble ∗ch, gdouble ∗rs)

    Extract the rotation angles for boxes on interaction list 4.
- gint **wbfmm_shift_angle_table_init** (void)

    Initialize table of angles for shift operations.
- **wbfmm_shift_operators_t** ∗ **wbfmm_shift_operators_new** (guint L, gdouble ∗work)

    Allocate shift operators and initialize rotations.
- gint **wbfmm_child_parent_shift_f** (gfloat ∗Cp, gint Np, gfloat ∗Cc, gint Nc, gfloat ∗H03, gfloat ∗H47, gint Lh, gfloat ∗shift, gint Ls, gfloat ∗work)

    Upward shift of singular expansion from eight children to common parent.
- gint **wbfmm_parent_child_shift_f** (gfloat ∗Cc, gint Nc, gfloat ∗Cp, gint Np, gfloat ∗H03, gfloat ∗H47, gint Lh, gfloat ∗shift, gint Ls, gfloat ∗work)

    Downward shift of parent expansion to child box centres.
- gint **wbfmm_shift_angles_list4_f** (gint i, gint j, gint k, gfloat ∗th, gfloat ∗ph, gfloat ∗ch, gfloat ∗rs)

    Extract the rotation angles for boxes on interaction list 4.
- gint **wbfmm_shift_angle_table_init_f** (void)

    Initialize table of angles for shift operations.
- **wbfmm_shift_operators_t** ∗ **wbfmm_shift_operators_new_f** (guint L, gfloat ∗work)

    Allocate shift operators and initialize rotations.

5.2.1 Detailed Description

Shift operations (combined rotation and translation) for upward and downward passes, and same-level interactions.

5.2.2 Function Documentation

5.2.2.1 gint wbfmm_child_parent_shift (gdouble ∗ *Cp,* gint *Np,* gdouble ∗ *Cc,* gint *Nc,* gdouble ∗ *H03,* gdouble ∗ *H47,* gint *Lh,* gdouble ∗ *trans,* gint *Ls,* gdouble ∗ *work*)

Upward shift of singular expansion from eight children to common parent.

Shift the expansion of eight child boxes to their parent and sum into the parent expansion. This function assumes data are packed with a stride of eight elements so that all expansion coefficients of a given order are contiguous in memory, ordered by Morton index.

Parameters

Cp	parent expansion array;

Np	order of parent expansion;
Cc	child expansion array;
Nc	order of child expansions;
H03	rotation coefficients for ‘lower' children (Morton index 0-3);
H47	rotation coefficients for ‘upper' children (Morton index 4-7);
Lh	maximum order of rotation coefficients;
trans	coaxial translation operator for distance between child and parent box centres;
Ls	order of *trans*;
work	workspace

Returns

0 on success

5.2.2.2 gint wbfmm_child_parent_shift_f (gfloat ∗ *Cp*, gint *Np*, gfloat ∗ *Cc*, gint *Nc*, gfloat ∗ *H03*, gfloat ∗ *H47*, gint *Lh*, gfloat ∗ *trans*, gint *Ls*, gfloat ∗ *work*)

Upward shift of singular expansion from eight children to common parent.

Shift the expansion of eight child boxes to their parent and sum into the parent expansion. This function assumes data are packed with a stride of eight elements so that all expansion coefficients of a given order are contiguous in memory, ordered by Morton index.

Parameters

Cp	parent expansion array;
Np	order of parent expansion;
Cc	child expansion array;
Nc	order of child expansions;
H03	rotation coefficients for ‘lower' children (Morton index 0-3);
H47	rotation coefficients for ‘upper' children (Morton index 4-7);
Lh	maximum order of rotation coefficients;
trans	coaxial translation operator for distance between child and parent box centres;
Ls	order of *trans*;
work	workspace

Returns

0 on success

5.2.2.3 gint wbfmm_parent_child_shift (gdouble ∗ *Cc*, gint *Nc*, gdouble ∗ *Cp*, gint *Np*, gdouble ∗ *H03*, gdouble ∗ *H47*, gint *Lh*, gdouble ∗ *trans*, gint *Ls*, gdouble ∗ *work*)

Downward shift of parent expansion to child box centres.

Shift the (regular) expansion data from a parent box to each of its child boxes, assuming the same packing as in **wbfmm_child_parent_shift** (p. 16)(...). Note that the rotation matrices for this function are switched relative to the rotations of the same name in **wbfmm_child_parent_shift** (p. 16)(...), because the ‘upper' children rotate ‘down' to be shifted to the parent centre but the rotation is ‘up' to shift from the parent to those children, and similarly for the ‘lower' children.

Parameters

Cc	child expansion array;
Nc	order of child expansions;
Cp	parent expansion array;
Np	order of parent expansion;
H03	rotation coefficients for ‘lower' children (Morton index 0-3);
H47	rotation coefficients for ‘upper' children (Morton index 4-7);
Lh	maximum order of rotation coefficients;
trans	coaxial translation operator for distance between child and parent box centres;
Ls	order of *trans*;
work	workspace

Returns

0 on success

5.2.2.4 gint wbfmm_parent_child_shift_f (gfloat ∗ *Cc*, gint *Nc*, gfloat ∗ *Cp*, gint *Np*, gfloat ∗ *H03*, gfloat ∗ *H47*, gint *Lh*, gfloat ∗ *trans*, gint *Ls*, gfloat ∗ *work*)

Downward shift of parent expansion to child box centres.

Shift the (regular) expansion data from a parent box to each of its child boxes, assuming the same packing as in **wbfmm_child_parent_shift_f** (p. 17)(...). Note that the rotation matrices for this function are switched relative to the rotations of the same name in **wbfmm_child_parent_shift_f** (p. 17)(...), because the ‘upper' children rotate ‘down' to be shifted to the parent centre but the rotation is ‘up' to shift from the parent to those children, and similarly for the ‘lower' children.

Parameters

Cc	child expansion array;
Nc	order of child expansions;
Cp	parent expansion array;
Np	order of parent expansion;
H03	rotation coefficients for ‘lower' children (Morton index 0-3);
H47	rotation coefficients for ‘upper' children (Morton index 4-7);
Lh	maximum order of rotation coefficients;
trans	coaxial translation operator for distance between child and parent box centres;
Ls	order of *trans*;
work	workspace

Returns

0 on success

5.2.2.5 gint wbfmm_shift_angle_table_init (void)

Initialize table of angles for shift operations.

This function must be called before any interaction calculations are performed, in particular before any call to **wbfmm_shift_operators_new** (p. 19)(...), in order to initialize the look-up table of orientations between boxes in interaction lists.

Returns

0 on success

5.2.2.6 gint wbfmm_shift_angle_table_init_f (void)

Initialize table of angles for shift operations.

This function must be called before any interaction calculations are performed, in particular before any call to **wbfmm_shift_operators_new_f** (p. 20)(...), in order to initialize the look-up table of orientations between boxes in interaction lists.

Returns

0 on success

5.2.2.7 gint wbfmm_shift_angles_list4 (gint *i*, gint *j*, gint *k*, gdouble $*$ *th*, gdouble $*$ *ph*, gdouble $*$ *ch*, gdouble $*$ *rs*)

Extract the rotation angles for boxes on interaction list 4.

Find the rotation angles $(\theta, \phi\,\chi)$ between a box at integer coordinates (i, j, k), using a look-up table which should be initialized with **wbfmm_shift_angle_table_init** (p. 18)(...)

Parameters

i	integer x coordinate of box on interaction list;
j	integer y coordinate of box on interaction list;
k	integer z coordinate of box on interaction list;
th	θ for rotation between boxes;
ph	ϕ for rotation between boxes;
ch	χ for rotation between boxes;
rs	scaling factor for distance between box centres, distance is *rs* multiplied by box width.

Returns

0 on success

5.2.2.8 gint wbfmm_shift_angles_list4_f (gint *i*, gint *j*, gint *k*, gfloat $*$ *th*, gfloat $*$ *ph*, gfloat $*$ *ch*, gfloat $*$ *rs*)

Extract the rotation angles for boxes on interaction list 4.

Find the rotation angles $(\theta, \phi\,\chi)$ between a box at integer coordinates (i, j, k), using a look-up table which should be initialized with **wbfmm_shift_angle_table_init_f** (p. 19)(...)

Parameters

i	integer x coordinate of box on interaction list;
j	integer y coordinate of box on interaction list;
k	integer z coordinate of box on interaction list;
th	θ for rotation between boxes;
ph	ϕ for rotation between boxes;
ch	χ for rotation between boxes;
rs	scaling factor for distance between box centres, distance is *rs* multiplied by box width.

Returns

0 on success

5.2.2.9 wbfmm_shift_operators_t $*$ wbfmm_shift_operators_new (guint *L*, gdouble $*$ *work*)

Allocate shift operators and initialize rotations.

Allocate a new **wbfmm_shift_operators_t** (p. 47) of given maximum order and initialize the rotation coefficients needed for same-level interaction calculations and upward and downward passes.

Parameters

L	maximum order of expansions;
work	workspace.

Returns

0 on success

5.2.2.10 wbfmm_shift_operators_t ∗ wbfmm_shift_operators_new_f (guint *L*, gfloat ∗ *work*)

Allocate shift operators and initialize rotations.

Allocate a new **wbfmm_shift_operators_t** (p. 47) of given maximum order and initialize the rotation coefficients needed for same-level interaction calculations and upward and downward passes.

Parameters

L	maximum order of expansions;
work	workspace.

Returns

0 on success

5.3 Generation and evaluation of expansions

Generation of regular and singular expansions and evaluation of them at field points.

Functions

- gint **wbfmm_expansion_h_cfft** (gdouble k, gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗q, gdouble ∗cfft, gint cstr, gdouble ∗work)

  Generation of singular expansion coefficients for point source.
- gint **wbfmm_expansion_dipole_h_cfft** (gdouble k, gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗fx, gdouble ∗fy, gdouble ∗fz, gdouble ∗cfft, gint cstr, gdouble ∗work)

  Generation of singular expansion coefficients for point dipole source.
- gint **wbfmm_expansion_h_evaluate** (gdouble k, gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

  Evaluate a singular expansion.
- gint **wbfmm_expansion_j_evaluate** (gdouble k, gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

  Evaluate a regular expansion.
- gint **wbfmm_expansion_h_cfft_f** (gfloat k, gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗q, gfloat ∗cfft, gint cstr, gfloat ∗work)

  Generation of singular expansion coefficients for point source.
- gint **wbfmm_expansion_dipole_h_cfft_f** (gfloat k, gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗fx, gfloat ∗fy, gfloat ∗fz, gfloat ∗cfft, gint cstr, gfloat ∗work)

  Generation of singular expansion coefficients for point dipole source.
- gint **wbfmm_expansion_h_evaluate_f** (gfloat k, gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

  Evaluate a singular expansion.
- gint **wbfmm_expansion_j_evaluate_f** (gfloat k, gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

  Evaluate a regular expansion.

5.3.1 Detailed Description

Generation of regular and singular expansions and evaluation of them at field points.

The functions described here handle spherical harmonic expansions of complex variables, solutions of the Helmholtz equations. Expansions of real variables are dealt with in a separate set of functions, for the Laplace equation (**Evaluation of the Laplace potential** (p. 41)). The expansion coefficients are packed in single- or double-precision arrays with the index of coefficient C_n^m, $-n \leq m \leq n$ given by $i = n(n+1)+m$. Coefficients are represented as real and imaginary parts, so that the coefficient is given by array entries $C_{si+0} + jC_{si+1}$ where i is the index, s is a stride allowing interleaved packing of data, and C_n is an array entry.

5.3.2 Function Documentation

5.3.2.1 gint wbfmm_expansion_dipole_h_cfft (gdouble *k*, gint *N*, gdouble ∗ *x0*, gdouble ∗ *xs*, gdouble ∗ *fx*, gdouble ∗ *fy*, gdouble ∗ *fz*, gdouble ∗ *cfft*, gint *cstr*, gdouble ∗ *work*)

Generation of singular expansion coefficients for point dipole source.

Parameters

k	wavenumber;
N	order of expansion;
x0	centre of expansion;
xs	source position;
fx	component of complex source strength;
fy	component of complex source strength;
fz	component of complex source strength;
cfft	incremented with expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
work	workspace

Returns

0 on success

5.3.2.2 gint wbfmm_expansion_dipole_h_cfft_f (gfloat *k*, gint *N*, gfloat ∗ *x0*, gfloat ∗ *xs*, gfloat ∗ *fx*, gfloat ∗ *fy*, gfloat ∗ *fz*, gfloat ∗ *cfft*, gint *cstr*, gfloat ∗ *work*)

Generation of singular expansion coefficients for point dipole source.

Parameters

k	wavenumber;
N	order of expansion;
x0	centre of expansion;
xs	source position;
fx	component of complex source strength;
fy	component of complex source strength;
fz	component of complex source strength;
cfft	incremented with expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
work	workspace

Returns

0 on success

5.3.2.3 gint wbfmm_expansion_h_cfft (gdouble *k*, gint *N*, gdouble ∗ *x0*, gdouble ∗ *xs*, gdouble ∗ *q*, gdouble ∗ *cfft*, gint *cstr*, gdouble ∗ *work*)

Generation of singular expansion coefficients for point source.

Parameters

k	wavenumber;
N	order of expansion;
x0	centre of expansion;
xs	source position;
q	complex source strength;
cfft	incremented with expansion coefficients;

cstr	stride in *cfft*, in number of complex elements;
work	workspace

Returns

0 on success

5.3.2.4 gint wbfmm_expansion_h_cfft_f (gfloat *k*, gint *N*, gfloat ∗ *x0*, gfloat ∗ *xs*, gfloat ∗ *q*, gfloat ∗ *cfft*, gint *cstr*, gfloat ∗ *work*)

Generation of singular expansion coefficients for point source.

Parameters

k	wavenumber;
N	order of expansion;
x0	centre of expansion;
xs	source position;
q	complex source strength;
cfft	incremented with expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
work	workspace

Returns

0 on success

5.3.2.5 gint wbfmm_expansion_h_evaluate (gdouble *k*, gdouble ∗ *x0*, gdouble ∗ *cfft*, gint *cstr*, gint *N*, gdouble ∗ *xf*, gdouble ∗ *field*, gdouble ∗ *work*)

Evaluate a singular expansion.

Parameters

k	wavenumber;
x0	centre of expansion;
cfft	expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
N	order of expansion;
xf	field point;
field	incremented with computed field;
work	workspace

Returns

0 on success

5.3.2.6 gint wbfmm_expansion_h_evaluate_f (gfloat *k*, gfloat ∗ *x0*, gfloat ∗ *cfft*, gint *cstr*, gint *N*, gfloat ∗ *xf*, gfloat ∗ *field*, gfloat ∗ *work*)

Evaluate a singular expansion.

Parameters

k	wavenumber;
x0	centre of expansion;
cfft	expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
N	order of expansion;
xf	field point;
field	incremented with computed field;
work	workspace

Returns

0 on success

5.3.2.7 gint wbfmm_expansion_j_evaluate (gdouble *k*, gdouble ∗ *x0*, gdouble ∗ *cfft*, gint *cstr*, gint *N*, gdouble ∗ *xf*, gdouble ∗ *field*, gdouble ∗ *work*)

Evaluate a regular expansion.

Parameters

k	wavenumber;
x0	centre of expansion;
cfft	expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
N	order of expansion;
xf	field point;
field	incremented with computed field;
work	workspace

Returns

0 on success

5.3.2.8 gint wbfmm_expansion_j_evaluate_f (gfloat *k*, gfloat ∗ *x0*, gfloat ∗ *cfft*, gint *cstr*, gint *N*, gfloat ∗ *xf*, gfloat ∗ *field*, gfloat ∗ *work*)

Evaluate a regular expansion.

Parameters

k	wavenumber;
x0	centre of expansion;
cfft	expansion coefficients;
cstr	stride in *cfft*, in number of complex elements;
N	order of expansion;
xf	field point;
field	incremented with computed field;
work	workspace

Returns

0 on success

5.4 Upward and downward passes

Upward and downward pass operations in octrees.

Functions

- gint **wbfmm_downward_pass** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gdouble ∗work)

    Perform downward pass at one level of an octree.
- gint **wbfmm_upward_pass** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gdouble ∗work)

    Perform upward pass at one level of an octree.
- gint **wbfmm_downward_pass_f** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gfloat ∗work)

    Perform downward pass at one level of an octree.
- gint **wbfmm_upward_pass_f** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gfloat ∗work)

    Perform upward pass at one level of an octree.

5.4.1 Detailed Description

Upward and downward pass operations in octrees.

5.4.2 Function Documentation

5.4.2.1 gint wbfmm_downward_pass (wbfmm_tree_t ∗ *t*, wbfmm_shift_operators_t ∗ *op*, guint *level*, gdouble ∗ *work*)

Perform downward pass at one level of an octree.

Perform one stage of a downward pass for tree levels greater than or equal to two. The actions performed are the evaluation of the list 4 contribution to the regular expansion and, for non-leaf boxes, a downward shift of the regular expansions to the child boxes at the next level.

Parameters

t	an initialized octree which has had the upward pass performed;
op	shift operators allocated for *t*;
level	level at which to perform downward pass;
work	workspace

Returns

0 on success

5.4.2.2 gint wbfmm_downward_pass_f (wbfmm_tree_t ∗ *t*, wbfmm_shift_operators_t ∗ *op*, guint *level*, gfloat ∗ *work*)

Perform downward pass at one level of an octree.

Perform one stage of a downward pass for tree levels greater than or equal to two. The actions performed are the evaluation of the list 4 contribution to the regular expansion and, for non-leaf boxes, a downward shift of the regular expansions to the child boxes at the next level.

Parameters

t	an initialized octree which has had the upward pass performed;
op	shift operators allocated for *t*;
level	level at which to perform downward pass;
work	workspace

Returns

0 on success

5.4.2.3 gint wbfmm_upward_pass (wbfmm_tree_t ∗ *t*, wbfmm_shift_operators_t ∗ *op*, guint *level*, gdouble ∗ *work*)

Perform upward pass at one level of an octree.

Perform one stage of the upward pass in an octree. The action performed is the upward shift of the singular expansions from boxes at level *level* to their parents.

Parameters

t	an initialized octree;
op	shift operators allocated for *t*;
level	level at which to perform upward pass;
work	workspace

Returns

0 on success

5.4.2.4 gint wbfmm_upward_pass_f (wbfmm_tree_t ∗ *t*, wbfmm_shift_operators_t ∗ *op*, guint *level*, gfloat ∗ *work*)

Perform upward pass at one level of an octree.

Perform one stage of the upward pass in an octree. The action performed is the upward shift of the singular expansions from boxes at level *level* to their parents.

Parameters

t	an initialized octree;
op	shift operators allocated for *t*;
level	level at which to perform upward pass;
work	workspace

Returns

0 on success

5.5 Rotation operators

Rotation of expansions.

Functions

- gint **wbfmm_rotation_angles** (gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗jx, gdouble ∗jy, gdouble ∗jz, gdouble ∗th, gdouble ∗ph, gdouble ∗ch)

  Compute the rotation angles (θ, ϕ, χ) *between axes.*
- gint **wbfmm_coefficients_H_rotation** (gdouble ∗H, gint N, gdouble th, gdouble ∗work)

  Compute rotation coefficients for angle θ*.*
- gint **wbfmm_rotation_angles_f** (gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗jx, gfloat ∗jy, gfloat ∗jz, gfloat ∗th, gfloat ∗ph, gfloat ∗ch)

  Compute the rotation angles (θ, ϕ, χ) *between axes.*
- gint **wbfmm_coefficients_H_rotation_f** (gfloat ∗H, gint N, gfloat th, gfloat ∗work)

  Compute rotation coefficients for angle θ*.*

5.5.1 Detailed Description

Rotation of expansions.

5.5.2 Function Documentation

5.5.2.1 gint wbfmm_coefficients_H_rotation (gdouble ∗ *H,* gint *N,* gdouble *th,* gdouble ∗ *work*)

Compute rotation coefficients for angle θ.

Generate the coefficients required to rotate one multipole expansion to a new orientation, using Gumerov and Duraiswami, Section 5, equation (5.48) and recursion (5.55). Coefficients *H* are real and densely packed on output.

Parameters

H	on output rotation coefficients;
N	maximum order of coefficients to compute;
th	rotation angle θ, from **wbfmm_rotation_angles** (p. 28)(...);
work	workspace.

Returns

0 on success

5.5.2.2 gint wbfmm_coefficients_H_rotation_f (gfloat ∗ *H,* gint *N,* gfloat *th,* gfloat ∗ *work*)

Compute rotation coefficients for angle θ.

Generate the coefficients required to rotate one multipole expansion to a new orientation, using Gumerov and Duraiswami, Section 5, equation (5.48) and recursion (5.55). Coefficients *H* are real and densely packed on output.

Parameters

H	on output rotation coefficients;

N	maximum order of coefficients to compute;
th	rotation angle θ, from **wbfmm_rotation_angles_f** (p. 28)(...);
work	workspace

Returns

0 on success

5.5.2.3 gint wbfmm_rotation_angles (gdouble ∗ *ix*, gdouble ∗ *iy*, gdouble ∗ *iz*, gdouble ∗ *jx*, gdouble ∗ *jy*, gdouble ∗ *jz*, gdouble ∗ *th*, gdouble ∗ *ph*, gdouble ∗ *ch*)

Compute the rotation angles (θ, ϕ, χ) between axes.

Compute the angles for rotation between two systems of axes $(\mathbf{i}_x, \mathbf{i}_y, \mathbf{i}_z)$ and $(\mathbf{j}_x, \mathbf{j}_y, \mathbf{j}_z)$, as defined in Section 5 of Gumerov and Duraiswami. All vectors should be unit length and form a right-handed coordinate system (no check is performed).

Parameters

ix	initial coordinate system x axis;
iy	initial coordinate system y axis;
iz	initial coordinate system z axis;
jx	rotated coordinate system x axis;
jy	rotated coordinate system y axis;
jz	rotated coordinate system z axis;
th	on exit, θ for rotation;
ph	on exit, ϕ for rotation;
ch	on exit, χ for rotation.

Returns

0 on success

5.5.2.4 gint wbfmm_rotation_angles_f (gfloat ∗ *ix*, gfloat ∗ *iy*, gfloat ∗ *iz*, gfloat ∗ *jx*, gfloat ∗ *jy*, gfloat ∗ *jz*, gfloat ∗ *th*, gfloat ∗ *ph*, gfloat ∗ *ch*)

Compute the rotation angles (θ, ϕ, χ) between axes.

Compute the angles for rotation between two systems of axes $(\mathbf{i}_x, \mathbf{i}_y, \mathbf{i}_z)$ and $(\mathbf{j}_x, \mathbf{j}_y, \mathbf{j}_z)$, as defined in Section 5 of Gumerov and Duraiswami. All vectors should be unit length and form a right-handed coordinate system (no check is performed).

Parameters

ix	initial coordinate system x axis;
iy	initial coordinate system y axis;
iz	initial coordinate system z axis;
jx	rotated coordinate system x axis;
jy	rotated coordinate system y axis;
jz	rotated coordinate system z axis;
th	on exit, θ for rotation;
ph	on exit, ϕ for rotation;

ch	on exit, χ for rotation.

Returns

0 on success

5.6 Translation operators

Translation of expansions.

Functions

- gint **wbfmm_coefficients_RR_coaxial** (gdouble ∗cfftRR, gint L, gdouble kr, gdouble ∗work)
  Generate coefficients for coaxial regular-to-regular translation.
- gint **wbfmm_coefficients_SR_coaxial** (gdouble ∗cfftSR, gint L, gdouble kr, gdouble ∗work)
  Generate coefficients for coaxial singular-to-regular translation.
- gint **wbfmm_coefficients_RR_coaxial_f** (gfloat ∗cfftRR, gint L, gfloat kr, gfloat ∗work)
  Generate coefficients for coaxial regular-to-regular translation.
- gint **wbfmm_coefficients_SR_coaxial_f** (gfloat ∗cfftSR, gint L, gfloat kr, gfloat ∗work)
  Generate coefficients for coaxial singular-to-regular translation.

5.6.1 Detailed Description

Translation of expansions.

5.6.2 Function Documentation

5.6.2.1 gint wbfmm_coefficients_RR_coaxial (gdouble ∗ *cfftRR,* gint *L,* gdouble *kr,* gdouble ∗ *work*)

Generate coefficients for coaxial regular-to-regular translation.

Generate translation coefficients for a regular-to-regular coaxial shift along the z axis of the local coordinate system, by distance r for wavenumber k, using the methods of Section 4.8 of Gumerov and Duraiswami. The regular-to-regular translation coefficients are identical to the singular-to-singular coefficients and are real.

Parameters

cfftRR	on output contains (real) translation coefficients;
L	maximum order of multipole expansion to be translated;
kr	coaxial translation parameter (wavenumber times distance);
work	workspace

Returns

0 on success

5.6.2.2 gint wbfmm_coefficients_RR_coaxial_f (gfloat ∗ *cfftRR,* gint *L,* gfloat *kr,* gfloat ∗ *work*)

Generate coefficients for coaxial regular-to-regular translation.

Generate translation coefficients for a regular-to-regular coaxial shift along the z axis of the local coordinate system, by distance r for wavenumber k, using the methods of Section 4.8 of Gumerov and Duraiswami. The regular-to-regular translation coefficients are identical to the singular-to-singular coefficients and are real.

Parameters

cfftRR	on output contains (real) translation coefficients;

L	maximum order of multipole expansion to be translated;
kr	coaxial translation parameter (wavenumber times distance);
work	workspace

Returns

0 on success

5.6.2.3 gint wbfmm_coefficients_SR_coaxial (gdouble $*$ *cfftSR*, gint *L*, gdouble *kr*, gdouble $*$ *work*)

Generate coefficients for coaxial singular-to-regular translation.

Generate translation coefficients for a singular-to-regular coaxial shift along the z axis of the local coordinate system, by distance r for wavenumber k, using the methods of Section 4.8 of Gumerov and Duraiswami. The output coefficients are complex.

Parameters

cfftSR	on output contains (complex) translation coefficients;
L	maximum order of multipole expansion to be translated;
kr	coaxial translation parameter (wavenumber times distance);
work	workspace

Returns

0 on success

5.6.2.4 gint wbfmm_coefficients_SR_coaxial_f (gfloat $*$ *cfftSR*, gint *L*, gfloat *kr*, gfloat $*$ *work*)

Generate coefficients for coaxial singular-to-regular translation.

Generate translation coefficients for a singular-to-regular coaxial shift along the z axis of the local coordinate system, by distance r for wavenumber k, using the methods of Section 4.8 of Gumerov and Duraiswami. The output coefficients are complex.

Parameters

cfftSR	on output contains (complex) translation coefficients;
L	maximum order of multipole expansion to be translated;
kr	coaxial translation parameter (wavenumber times distance);
work	workspace

Returns

0 on success

5.7 Utility and convenience functions

Various functions of use in debugging or underlying utilities.

Functions

- gint **wbfmm_legendre_recursion_array** (gdouble ∗∗Pnm1, gdouble ∗∗Pn, gint n, gdouble C, gdouble S)

  Perform recursion on normalized associated Legendre functions.
- gint **wbfmm_legendre_init** (gdouble C, gdouble S, gdouble ∗P0, gdouble ∗P10, gdouble ∗P11)

  Initialize normalized associated Legendre functions.
- gint **wbfmm_bessel_j_recursion** (gdouble ∗jnm1, gdouble ∗jn, gdouble x, gint n)

  Perform recursion on spherical Bessel function $j_n(x)$.
- gint **wbfmm_bessel_j_init** (gdouble x, gdouble ∗j0, gdouble ∗j1)

  Initialize the spherical Bessel function recursion.
- gint **wbfmm_bessel_h_init** (gdouble x, gdouble ∗h0, gdouble ∗h1)

  Initialize spherical Hankel function recursion.
- gint **wbfmm_bessel_h_recursion** (gdouble ∗hnm1, gdouble ∗hn, gdouble x, gint n)

  Perform one step of spherical Hankel recursion.
- gint **wbfmm_total_dipole_field** (gdouble k, gdouble ∗xs, gint xstride, gdouble ∗src, gint sstride, gint nsrc, gdouble ∗xf, gdouble ∗field)

  Compute total field from dipole sources by direct evaluation.
- gint **wbfmm_coordinate_transform** (gdouble ∗x, gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗y)

  Transform coordinates to rotated axes.
- gint **wbfmm_shift_coordinates** (gdouble ∗x, gdouble ∗y, gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗r)

  Find system of axes for coordinate shift.
- gint **wbfmm_box_location_from_index** (guint64 i, guint32 level, gdouble ∗x0, gdouble D, gdouble ∗x, gdouble ∗wb)

  Find the coordinates of a box from its Morton index.
- gint **wbfmm_shift_angles** (gdouble ∗xi, gdouble ∗xj, gdouble ∗th, gdouble ∗ph, gdouble ∗ch, gdouble ∗r)

  Compute angles and distance to shift expansion between two points.
- gint **wbfmm_tree_write_sources** (**wbfmm_tree_t** ∗t, gdouble ∗q, gint stride, FILE ∗f)

  Write a tree source list to file.
- gint **wbfmm_legendre_recursion_array_f** (gfloat ∗∗Pnm1, gfloat ∗∗Pn, gint n, gfloat C, gfloat S)

  Perform recursion on normalized associated Legendre functions.
- gint **wbfmm_legendre_init_f** (gfloat C, gfloat S, gfloat ∗P0, gfloat ∗P10, gfloat ∗P11)

  Initialize normalized associated Legendre functions.
- gint **wbfmm_bessel_j_recursion_f** (gfloat ∗jnm1, gfloat ∗jn, gfloat x, gint n)

  Perform recursion on spherical Bessel function $j_n(x)$.
- gint **wbfmm_bessel_j_init_f** (gfloat x, gfloat ∗j0, gfloat ∗j1)

  Initialize the spherical Bessel function recursion.
- gint **wbfmm_bessel_h_init_f** (gfloat x, gfloat ∗h0, gfloat ∗h1)

  Initialize spherical Hankel function recursion.
- gint **wbfmm_bessel_h_recursion_f** (gfloat ∗hnm1, gfloat ∗hn, gfloat x, gint n)

  Perform one step of spherical Hankel recursion.
- gint **wbfmm_total_dipole_field_f** (gfloat k, gfloat ∗xs, gint xstride, gfloat ∗src, gint sstride, gint nsrc, gfloat ∗xf, gfloat ∗field)

  Compute total field from dipole sources by direct evaluation.
- gint **wbfmm_coordinate_transform_f** (gfloat ∗x, gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗y)

  Transform coordinates to rotated axes.
- gint **wbfmm_shift_coordinates_f** (gfloat ∗x, gfloat ∗y, gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗r)

  Find system of axes for coordinate shift.

- gint **wbfmm_box_location_from_index_f** (guint64 i, guint32 level, gfloat ∗x0, gfloat D, gfloat ∗x, gfloat ∗wb)

  Find the coordinates of a box from its Morton index.
- gint **wbfmm_shift_angles_f** (gfloat ∗xi, gfloat ∗xj, gfloat ∗th, gfloat ∗ph, gfloat ∗ch, gfloat ∗r)

  Compute angles and distance to shift expansion between two points.
- gint **wbfmm_tree_write_sources_f** (**wbfmm_tree_t** ∗t, gfloat ∗q, gint stride, FILE ∗f)

  Write a tree source list to file.

5.7.1 Detailed Description

Various functions of use in debugging or underlying utilities.

5.7.2 Function Documentation

5.7.2.1 gint wbfmm_bessel_h_init (gdouble *x*, gdouble ∗ *h0*, gdouble ∗ *h1*)

Initialize spherical Hankel function recursion.

Parameters

x	argument of $h_n(x)$;
h0	on exit $h_0(x)$;
h1	on exit $h_1(x)$

Returns

0 on success

5.7.2.2 gint wbfmm_bessel_h_init_f (gfloat *x*, gfloat ∗ *h0*, gfloat ∗ *h1*)

Initialize spherical Hankel function recursion.

Parameters

x	argument of $h_n(x)$;
h0	on exit $h_0(x)$;
h1	on exit $h_1(x)$

Returns

0 on success

5.7.2.3 gint wbfmm_bessel_h_recursion (gdouble ∗ *hnm1*, gdouble ∗ *hn*, gdouble *x*, gint *n*)

Perform one step of spherical Hankel recursion.

Perform one step of the spherical Hankel function recursion. On entry *hnm1* and *hnm* contain $h_{n-1}(x)$ and $h_n(x)$ respectively. On exit they contain equivalent values but for n incremented by one. When x falls below a small order-dependent cutoff, where the recursion is unreliable, $h_n(x)$ is computed directly using a power series.

Parameters

hnm1	$h_{n-1}(x)$;
hn	$h_n(x)$;
x	argument of spherical Hankel function;
n	order of spherical Hankel function

Returns

0 on success

5.7.2.4 gint wbfmm_bessel_h_recursion_f (gfloat ∗ *hnm1*, gfloat ∗ *hn*, gfloat *x*, gint *n*)

Perform one step of spherical Hankel recursion.

Perform one step of the spherical Hankel function recursion. On entry *hnm1* and *hnm* contain $h_{n-1}(x)$ and $h_n(x)$ respectively. On exit they contain equivalent values but for n incremented by one. When x falls below a small order-dependent cutoff, where the recursion is unreliable, $h_n(x)$ is computed directly using a power series.

Parameters

hnm1	$h_{n-1}(x)$;
hn	$h_n(x)$;
x	argument of spherical Hankel function;
n	order of spherical Hankel function

Returns

0 on success

5.7.2.5 gint wbfmm_bessel_j_init (gdouble *x*, gdouble ∗ *j0*, gdouble ∗ *j1*)

Initialize the spherical Bessel function recursion.

Parameters

x	argument of $j_n(x)$;
j0	on exit $j_0(x)$;
j1	on exit $j_1(x)$

Returns

0 on success

5.7.2.6 gint wbfmm_bessel_j_init_f (gfloat *x*, gfloat ∗ *j0*, gfloat ∗ *j1*)

Initialize the spherical Bessel function recursion.

Parameters

x	argument of $j_n(x)$;
j0	on exit $j_0(x)$;
j1	on exit $j_1(x)$

Returns

0 on success

5.7.2.7 gint wbfmm_bessel_j_recursion (gdouble * *jnm1*, gdouble * *jn*, gdouble *x*, gint *n*)

Perform recursion on spherical Bessel function $j_n(x)$.

Perform one step of the spherical Bessel function recursion. On entry *jnm1* and *jnm* contain $j_{n-1}(x)$ and $j_n(x)$ respectively. On exit they contain equivalent values but for n incremented by one. When x falls below a small order-dependent cutoff, where the recursion is unreliable, $j_n(x)$ is computed directly using a power series.

Parameters

jnm1	$j_{n-1}(x)$;
jn	$j_n(x)$;
x	argument of spherical Bessel function;
n	order of spherical Bessel function

Returns

0 on success

5.7.2.8 gint wbfmm_bessel_j_recursion_f (gfloat * *jnm1*, gfloat * *jn*, gfloat *x*, gint *n*)

Perform recursion on spherical Bessel function $j_n(x)$.

Perform one step of the spherical Bessel function recursion. On entry *jnm1* and *jnm* contain $j_{n-1}(x)$ and $j_n(x)$ respectively. On exit they contain equivalent values but for n incremented by one. When x falls below a small order-dependent cutoff, where the recursion is unreliable, $j_n(x)$ is computed directly using a power series.

Parameters

jnm1	$j_{n-1}(x)$;
jn	$j_n(x)$;
x	argument of spherical Bessel function;
n	order of spherical Bessel function

Returns

0 on success

5.7.2.9 gint wbfmm_box_location_from_index (guint64 *idx*, guint32 *level*, gdouble * *x0*, gdouble *D*, gdouble * *x*, gdouble * *wb*)

Find the coordinates of a box from its Morton index.

Parameters

idx	Morton index of box;
level	level in octree of box;
x0	origin of top-level box;
D	width of top-level box;
x	coordinates of box *idx* at level *level*;
wb	width of box at level *level*

Returns

0 on success

5.7.2.10 gint wbfmm_box_location_from_index_f (guint64 *idx*, guint32 *level*, gfloat * *x0*, gfloat *D*, gfloat * *x*, gfloat * *wb*)

Find the coordinates of a box from its Morton index.

Parameters

idx	Morton index of box;
level	level in octree of box;
x0	origin of top-level box;
D	width of top-level box;
x	coordinates of box *idx* at level *level*;
wb	width of box at level *level*

Returns

0 on success

5.7.2.11 gint wbfmm_coordinate_transform (gdouble ∗ *x*, gdouble ∗ *ix*, gdouble ∗ *iy*, gdouble ∗ *iz*, gdouble ∗ *y*)

Transform coordinates to rotated axes.

Parameters

x	point coordinates in original axes;
ix	unit vector in new axes;
iy	unit vector in new axes;
iz	unit vector in new axes;
y	point coordinates in new axes

Returns

0 on success

5.7.2.12 gint wbfmm_coordinate_transform_f (gfloat ∗ *x*, gfloat ∗ *ix*, gfloat ∗ *iy*, gfloat ∗ *iz*, gfloat ∗ *y*)

Transform coordinates to rotated axes.

Parameters

x	point coordinates in original axes;
ix	unit vector in new axes;
iy	unit vector in new axes;
iz	unit vector in new axes;
y	point coordinates in new axes

Returns

0 on success

5.7.2.13 gint wbfmm_legendre_init (gdouble *C*, gdouble *S*, gdouble ∗ *P0*, gdouble ∗ *P10*, gdouble ∗ *P11*)

Initialize normalized associated Legendre functions.

Parameters

C	$\cos\theta$;

S	$\sin\theta$;
P0	on output $P_0^0(\cos\theta)$;
P10	on output $P_1^0(\cos\theta)$;
P11	on output $P_1^1(\cos\theta)$;

Returns

0 on success

5.7.2.14 gint wbfmm_legendre_init_f (gfloat *C*, gfloat *S*, gfloat $*$ *P0*, gfloat $*$ *P10*, gfloat $*$ *P11*)

Initialize normalized associated Legendre functions.

Parameters

C	$\cos\theta$;
S	$\sin\theta$;
P0	on output $P_0^0(\cos\theta)$;
P10	on output $P_1^0(\cos\theta)$;
P11	on output $P_1^1(\cos\theta)$;

Returns

0 on success

5.7.2.15 gint wbfmm_legendre_recursion_array (gdouble $**$ *Pnm1*, gdouble $**$ *Pn*, gint *n*, gdouble *C*, gdouble *S*)

Perform recursion on normalized associated Legendre functions.

Perform recursion on normalized associated Legendre functions with input $P_{n-1}^m(\cos\theta)$, $0 \leq m \leq n-1$, and $P_n^m(\cos\theta)$, $0 \leq m \leq n$, generating equivalent outputs with n incremented by one. Note that the arrays of associated Legendre functions are switched internally to ensure that the ordering remains correct after the recursion step.

Parameters

Pnm1	pointer to array of normalized associated Legendre functions for $n-1$;
Pn	pointer to array of normalized associated Legendre functions for n;
n	order of *Pn*;
C	$\cos\theta$;
S	$\sin\theta$;

Returns

0 on success

5.7.2.16 gint wbfmm_legendre_recursion_array_f (gfloat $**$ *Pnm1*, gfloat $**$ *Pn*, gint *n*, gfloat *C*, gfloat *S*)

Perform recursion on normalized associated Legendre functions.

Perform recursion on normalized associated Legendre functions with input $P_{n-1}^m(\cos\theta)$, $0 \leq m \leq n-1$, and $P_n^m(\cos\theta)$, $0 \leq m \leq n$, generating equivalent outputs with n incremented by one. Note that the arrays of associated Legendre functions are switched internally to ensure that the ordering remains correct after the recursion step.

Parameters

Pnm1	pointer to array of normalized associated Legendre functions for $n-1$;
Pn	pointer to array of normalized associated Legendre functions for n;
n	order of *Pn*;
C	$\cos\theta$;
S	$\sin\theta$;

Returns

0 on success

5.7.2.17 gint wbfmm_shift_angles (gdouble $*$ *xi*, gdouble $*$ *xj*, gdouble $*$ *th*, gdouble $*$ *ph*, gdouble $*$ *ch*, gdouble $*$ *r*)

Compute angles and distance to shift expansion between two points.

This is a combination of a call to **wbfmm_shift_coordinates** (p. 38)(...) and **wbfmm_rotation_angles** (p. 28)(...)

Parameters

xi	origin of shift;
xj	destination of shift;
th	θ for shift;
ph	ϕ for shift;
ch	χ for shift;
r	distance between source and destination points

Returns

0 on success

5.7.2.18 gint wbfmm_shift_angles_f (gfloat $*$ *xi*, gfloat $*$ *xj*, gfloat $*$ *th*, gfloat $*$ *ph*, gfloat $*$ *ch*, gfloat $*$ *r*)

Compute angles and distance to shift expansion between two points.

This is a combination of a call to **wbfmm_shift_coordinates_f** (p. 39)(...) and **wbfmm_rotation_angles_↩ f** (p. 28)(...)

Parameters

xi	origin of shift;
xj	destination of shift;
th	θ for shift;
ph	ϕ for shift;
ch	χ for shift;
r	distance between source and destination points

Returns

0 on success

5.7.2.19 gint wbfmm_shift_coordinates (gdouble $*$ *x*, gdouble $*$ *y*, gdouble $*$ *ix*, gdouble $*$ *iy*, gdouble $*$ *iz*, gdouble $*$ *r*)

Find system of axes for coordinate shift.

Parameters

x	origin of shift;
y	point to shift to;
ix	on output unit vector of shift axes;
iy	on output unit vector of shift axes;
iz	on output unit vector of shift axes in direction of shift;
r	distance between two input points

Returns

0 on success

5.7.2.20 gint wbfmm_shift_coordinates_f (gfloat ∗ *x*, gfloat ∗ *y*, gfloat ∗ *ix*, gfloat ∗ *iy*, gfloat ∗ *iz*, gfloat ∗ *r*)

Find system of axes for coordinate shift.

Parameters

x	origin of shift;
y	point to shift to;
ix	on output unit vector of shift axes;
iy	on output unit vector of shift axes;
iz	on output unit vector of shift axes in direction of shift;
r	distance between two input points

Returns

0 on success

5.7.2.21 gint wbfmm_total_dipole_field (gdouble *k*, gdouble ∗ *xs*, gint *xstride*, gdouble ∗ *src*, gint *sstride*, gint *nsrc*, gdouble ∗ *xf*, gdouble ∗ *field*)

Compute total field from dipole sources by direct evaluation.

Evaluate the field at some point $\mathbf{x}$ by direct evaluation of the sum over sources at $\mathbf{x}_n$ $\sum_{n=1}^{N} \mathbf{f}_n.\nabla h_0(\mathbf{x}-\mathbf{x}_n)/4\pi$.

Parameters

k	wavenumber;
xs	array of source positions;
xstride	stride in *xs* between source positions;
src	array of complex vector source strengths;
sstride	stride in *src*;
nsrc	number of sources;
xf	point for field evaluation;
field	incremented with computed field

Returns

0 on success

5.7.2.22 gint wbfmm_total_dipole_field_f (gfloat *k*, gfloat ∗ *xs*, gint *xstride*, gfloat ∗ *src*, gint *sstride*, gint *nsrc*, gfloat ∗ *xf*, gfloat ∗ *field*)

Compute total field from dipole sources by direct evaluation.

Evaluate the field at some point $\mathbf{x}$ by direct evaluation of the sum over sources at $\mathbf{x}_n$ $\sum_{n=1}^{N} \mathbf{f}_n.\nabla h_0(\mathbf{x}-\mathbf{x}_n)/4\pi$.

Parameters

k	wavenumber;
xs	array of source positions;
xstride	stride in *xs* between source positions;
src	array of complex vector source strengths;
sstride	stride in *src*;
nsrc	number of sources;
xf	point for field evaluation;
field	incremented with computed field

Returns

0 on success

5.7.2.23 gint wbfmm_tree_write_sources (wbfmm_tree_t * *t*, gdouble * *q*, gint *stride*, FILE * *f*)

Write a tree source list to file.

Write to file a list of source positions attached to an octree, in order of Morton index by which they are attached to leaf boxes. If source strengths are supplied (*q* not NULL) these are also written to file.

Parameters

t	an octree with a list of sources attached;
q	source strengths (if NULL, source strengths are not written);
stride	source strength stride in *q*;
f	output file to write to

Returns

0 on success

5.7.2.24 gint wbfmm_tree_write_sources_f (wbfmm_tree_t * *t*, gfloat * *q*, gint *stride*, FILE * *f*)

Write a tree source list to file.

Write to file a list of source positions attached to an octree, in order of Morton index by which they are attached to leaf boxes. If source strengths are supplied (*q* not NULL) these are also written to file.

Parameters

t	an octree with a list of sources attached;
q	source strengths (if NULL, source strengths are not written);
stride	source strength stride in *q*;
f	output file to write to

Returns

0 on success

5.8 Evaluation of the Laplace potential

Variants on standard functions to allow WBFMM to be used for the Laplace equation.

Functions

- gint **wbfmm_expansion_laplace_cfft** (gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗q, gint nq, gdouble ∗cfft, gint cstr, gdouble ∗work)

  Generation of singular expansion coefficients for point source in Laplace problem.
- gint **wbfmm_expansion_laplace_evaluate** (gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gint nq, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

  Evaluate a singular expansion for Laplace problem.
- gint **wbfmm_coaxial_translate_SS_laplace** (gdouble ∗Co, gint cstro, gint No, gdouble ∗Ci, gint cstri, gint Ni, gint nq, gdouble t)

  Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.
- gint **wbfmm_expansion_laplace_cfft_f** (gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗q, gint nq, gfloat ∗cfft, gint cstr, gfloat ∗work)

  Generation of singular expansion coefficients for point source in Laplace problem.
- gint **wbfmm_expansion_laplace_evaluate_f** (gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gint nq, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

  Evaluate a singular expansion for Laplace problem.
- gint **wbfmm_coaxial_translate_SS_laplace_f** (gfloat ∗Co, gint cstro, gint No, gfloat ∗Ci, gint cstri, gint Ni, gint nq, gfloat t)

  Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.

5.8.1 Detailed Description

Variants on standard functions to allow WBFMM to be used for the Laplace equation.

5.8.2 Function Documentation

5.8.2.1 gint wbfmm_coaxial_translate_SS_laplace (gdouble ∗ *Co,* gint *cstro,* gint *No,* gdouble ∗ *Ci,* gint *cstri,* gint *Ni,* gint *nq,* gdouble *t*)

Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.

Parameters

Co	on output, expansion about new centre;
cstro	stride in *Co*;
No	order of expansion in *Co*;
Ci	input expansion coefficients;
cstri	stride in *Ci*;
Ni	order of expansion in *Ci*;
nq	number of source terms in expansion;
t	distance to translate expansion.

Returns

0 on success

5.8.2.2 gint wbfmm_coaxial_translate_SS_laplace_f (gfloat ∗ *Co,* gint *cstro,* gint *No,* gfloat ∗ *Ci,* gint *cstri,* gint *Ni,* gint *nq,* gfloat *t*)

Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.

Parameters

Co	on output, expansion about new centre;
cstro	stride in *Co*;
No	order of expansion in *Co*;
Ci	input expansion coefficients;
cstri	stride in *Ci*;
Ni	order of expansion in *Ci*;
nq	number of source terms in expansion;
t	distance to translate expansion.

Returns

0 on success

5.8.2.3 gint wbfmm_expansion_laplace_cfft (gint *N*, gdouble ∗ *x0*, gdouble ∗ *xs*, gdouble ∗ *q*, gint *nq*, gdouble ∗ *cfft*, gint *cstr*, gdouble ∗ *work*)

Generation of singular expansion coefficients for point source in Laplace problem.

Parameters

N	order of expansion;
x0	origin of expansion;
xs	source position;
q	source strength(s);
nq	number of components in *q*;
cfft	on output, incremented with coefficients of expansion;
cstr	stride in *cfft* (must be at least equal to *nq*);
work	workspace.

Returns

0 on success

5.8.2.4 gint wbfmm_expansion_laplace_cfft_f (gint *N*, gfloat ∗ *x0*, gfloat ∗ *xs*, gfloat ∗ *q*, gint *nq*, gfloat ∗ *cfft*, gint *cstr*, gfloat ∗ *work*)

Generation of singular expansion coefficients for point source in Laplace problem.

Parameters

N	order of expansion;
x0	origin of expansion;
xs	source position;
q	source strength(s);
nq	number of components in *q*;
cfft	on output, incremented with coefficients of expansion;
cstr	stride in *cfft* (must be at least equal to *nq*);
work	workspace.

Returns

0 on success

5.8.2.5 gint wbfmm_expansion_laplace_evaluate (gdouble ∗ *x0,* gdouble ∗ *cfft,* gint *cstr,* gint *N,* gint *nq,* gdouble ∗ *xf,* gdouble ∗ *field,* gdouble ∗ *work*)

Evaluate a singular expansion for Laplace problem.

Parameters

x0	origin of expansion;
cfft	on output, incremented with coefficients of expansion;
cstr	stride in *cfft* (must be at least equal to *nq*);
N	order of expansion;
nq	number of components in *q*;
xf	field point;
field	computed potential for each of the *nq* components;
work	workspace.

Returns

0 on success

5.8.2.6 gint wbfmm_expansion_laplace_evaluate_f (gfloat * *x0*, gfloat * *cfft*, gint *cstr*, gint *N*, gint *nq*, gfloat * *xf*, gfloat * *field*, gfloat * *work*)

Evaluate a singular expansion for Laplace problem.

Parameters

x0	origin of expansion;
cfft	on output, incremented with coefficients of expansion;
cstr	stride in *cfft* (must be at least equal to *nq*);
N	order of expansion;
nq	number of components in *q*;
xf	field point;
field	computed potential for each of the *nq* components;
work	workspace.

Returns

0 on success

5.9 Indexing and lookup operations

Indexing functions for accessing tree data structures.

Functions

- guint64 **wbfmm_box_index** (guint32 i, guint32 j, guint32 k)
- gint **wbfmm_box_location** (guint64 idx, guint32 ∗i, guint32 ∗j, guint32 ∗k)

5.9.1 Detailed Description

Indexing functions for accessing tree data structures.

Functions for indexing and lookup in tree data structures, including finding neighbours and interaction lists, based on the methods of Gumerov, Duraiswami, and Borovikov, Data Structures, Optimal Choice of Parameters, and Complexity Results for Generalized Multilevel Fast Multipole Methods in d Dimensions, 2003

`http://users.umiacs.umd.edu/~gumerov/PDFs/cs-tr-4458.pdf`

Code for Morton indexing operations is taken from: `https://www.forceflow.be/2013/10/07/morton-encodingdec`

5.9.2 Function Documentation

5.9.2.1 guint64 wbfmm_box_index (guint32 *i*, guint32 *j*, guint32 *k*)

Generate a Morton index for a box with corner at integer coordinates (i,j,k).

Parameters

i	x index of bottom left hand corner;
j	y index of bottom left hand corner;
k	z index of bottom left hand corner.

Returns

Morton index for (*i*, *j*, *k*).

5.9.2.2 gint wbfmm_box_location (guint64 *idx*, guint32 ∗ *i*, guint32 ∗ *j*, guint32 ∗ *k*)

Compute indices for bottom left hand corner of box defined by its Morton index, as generated by **wbfmm_box_index** (p. 46)

Parameters

idx	index of box corner;
i	on output, x index of bottom left hand corner of box;
j	on output, y index of bottom left hand corner of box;
k	on output, z index of bottom left hand corner of box.

Returns

0 on success.

Chapter 6

Data Structure Documentation

6.1 wbfmm_box_t Struct Reference

Data Fields

- guint32 **i**
- guint32 **n**
- gpointer **mps**
- gpointer **mpr**

6.1.1 Detailed Description

Data type for octree boxes

6.1.2 Field Documentation

6.1.2.1 guint32 wbfmm_box_t::i

index of first source point in box

6.1.2.2 gpointer wbfmm_box_t::mpr

pointer to regular multipole expansion data

6.1.2.3 gpointer wbfmm_box_t::mps

pointer to singular multipole expansion data

6.1.2.4 guint32 wbfmm_box_t::n

number of points in box

6.2 wbfmm_shift_operators_t Struct Reference

Data Fields

- gsize **size**
- guint **nlevels**
- guint **L** [WBFMM_TREE_MAX_DEPTH+1]
- guint **nerot**
- gpointer **SR** [WBFMM_TREE_MAX_DEPTH+1]
- gpointer **SS** [WBFMM_TREE_MAX_DEPTH+1]
- gpointer **rotations**

6.2.1 Detailed Description

Data type holding operators for upward and downward passes and interaction calculations at each level

6.2.2 Field Documentation

6.2.2.1 guint wbfmm_shift_operators_t::L[WBFMM_TREE_MAX_DEPTH+1]

$<$ number of levels in tree

6.2.2.2 guint wbfmm_shift_operators_t::nerot

$<$ maximum order of expansion per level

6.2.2.3 guint wbfmm_shift_operators_t::nlevels

$<$ maximum order of expansions

6.2.2.4 gpointer wbfmm_shift_operators_t::rotations

$<$ singular-to-singular (regular-to-regular) coaxial translations

6.2.2.5 gsize wbfmm_shift_operators_t::size

size of data type, i.e. float or double

6.2.2.6 gpointer wbfmm_shift_operators_t::SR[WBFMM_TREE_MAX_DEPTH+1]

$<$ number of elements in rotation operators

6.2.2.7 gpointer wbfmm_shift_operators_t::SS[WBFMM_TREE_MAX_DEPTH+1]

$<$ singular-to-regular coaxial translations

6.3 wbfmm_tree_t Struct Reference

Data Fields

- **wbfmm_box_t * boxes** [WBFMM_TREE_MAX_DEPTH+1]

- guint **maxpoints**
- guint **npoints**
- guint ∗ **ip**
- guint **depth**
- guint **order_s** [WBFMM_TREE_MAX_DEPTH+1]
- guint **order_r** [WBFMM_TREE_MAX_DEPTH+1]
- gchar **x** [24]
- gchar ∗ **points**
- gpointer ∗ **mps** [WBFMM_TREE_MAX_DEPTH+1]
- gpointer ∗ **mpr** [WBFMM_TREE_MAX_DEPTH+1]
- gsize **pstr**
- gdouble **D**

6.3.1 Detailed Description

Data type for octrees

6.3.2 Field Documentation

6.3.2.1 wbfmm_box_t∗ wbfmm_tree_t::boxes[WBFMM_TREE_MAX_DEPTH+1]

arrays of boxes at each level

Referenced by wbfmm_tree_add_level().

6.3.2.2 gdouble wbfmm_tree_t::D

width of domain cube

6.3.2.3 guint wbfmm_tree_t::depth

depth of tree

Referenced by wbfmm_tree_add_level().

6.3.2.4 guint ∗ wbfmm_tree_t::ip

indices of points, sorted by Morton index

6.3.2.5 guint wbfmm_tree_t::maxpoints

maximum number of points in tree

6.3.2.6 gpointer ∗ wbfmm_tree_t::mpr[WBFMM_TREE_MAX_DEPTH+1]

regular expansion data at each level

6.3.2.7 gpointer∗ wbfmm_tree_t::mps[WBFMM_TREE_MAX_DEPTH+1]

singular expansion data at each level

6.3.2.8 guint wbfmm_tree_t::npoints

number of points in tree

6.3.2.9 guint wbfmm_tree_t::order_r[WBFMM_TREE_MAX_DEPTH+1]

order of regular expansions at each level

6.3.2.10 guint wbfmm_tree_t::order_s[WBFMM_TREE_MAX_DEPTH+1]

order of singular expansions at each level

6.3.2.11 gchar ∗ wbfmm_tree_t::points

point coordinates

6.3.2.12 gsize wbfmm_tree_t::pstr

stride in point data

6.3.2.13 gchar wbfmm_tree_t::x[24]

origin of tree domain cube

Chapter 7

File Documentation

7.1 tree.c File Reference

Functions

- gint **wbfmm_tree_add_level** (**wbfmm_tree_t** ∗t)

7.1.1 Detailed Description

Author

Michael Carley `ensmjc@rpc-ensmjc.bath.ac.uk`

Date

Mon Jun 24 14:51:21 2019

7.2 wbfmm.h File Reference

Header for Wide Band FMM library.

Data Structures

- struct **wbfmm_box_t**
- struct **wbfmm_tree_t**
- struct **wbfmm_shift_operators_t**

Enumerations

- enum **wbfmm_problem_t** { **WBFMM_PROBLEM_LAPLACE** = 1, **WBFMM_PROBLEM_HELMHOLTZ** = 2 }

Functions

- gint **wbfmm_shift_coordinates** (gdouble ∗x, gdouble ∗y, gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗r)

  Find system of axes for coordinate shift.
- gint **wbfmm_legendre_recursion_array** (gdouble ∗∗Pnm1, gdouble ∗∗Pn, gint n, gdouble C, gdouble S)

Perform recursion on normalized associated Legendre functions.

- gint **wbfmm_bessel_j_recursion** (gdouble ∗jnm1, gdouble ∗jn, gdouble x, gint n)

    Perform recursion on spherical Bessel function $j_n(x)$.
- gint **wbfmm_bessel_h_recursion** (gdouble ∗hnm1, gdouble ∗hn, gdouble x, gint n)

    Perform one step of spherical Hankel recursion.
- gint **wbfmm_bessel_j_init** (gdouble x, gdouble ∗j0, gdouble ∗j1)

    Initialize the spherical Bessel function recursion.
- gint **wbfmm_bessel_h_init** (gdouble x, gdouble ∗h0, gdouble ∗h1)

    Initialize spherical Hankel function recursion.
- gint **wbfmm_legendre_init** (gdouble C, gdouble S, gdouble ∗P0, gdouble ∗P10, gdouble ∗P11)

    Initialize normalized associated Legendre functions.
- gint **wbfmm_expansion_h_cfft** (gdouble k, gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗q, gdouble ∗cfft, gint cstr, gdouble ∗work)

    Generation of singular expansion coefficients for point source.
- gint **wbfmm_expansion_dipole_h_cfft** (gdouble k, gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗fx, gdouble ∗fy, gdouble ∗fz, gdouble ∗cfft, gint cstr, gdouble ∗work)

    Generation of singular expansion coefficients for point dipole source.
- gint **wbfmm_expansion_h_evaluate** (gdouble k, gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

    Evaluate a singular expansion.
- gint **wbfmm_expansion_j_evaluate** (gdouble k, gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

    Evaluate a regular expansion.
- gint **wbfmm_total_dipole_field** (gdouble k, gdouble ∗xs, gint xstride, gdouble ∗src, gint sstride, gint nsrc, gdouble ∗xf, gdouble ∗field)

    Compute total field from dipole sources by direct evaluation.
- gint **wbfmm_coordinate_transform** (gdouble ∗x, gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗y)

    Transform coordinates to rotated axes.
- gint **wbfmm_coefficients_RR_coaxial** (gdouble ∗cfftRR, gint L, gdouble kr, gdouble ∗work)

    Generate coefficients for coaxial regular-to-regular translation.
- gint **wbfmm_coefficients_SR_coaxial** (gdouble ∗cfftSR, gint L, gdouble kr, gdouble ∗work)

    Generate coefficients for coaxial singular-to-regular translation.
- gint **wbfmm_rotation_angles** (gdouble ∗ix, gdouble ∗iy, gdouble ∗iz, gdouble ∗jx, gdouble ∗jy, gdouble ∗jz, gdouble ∗th, gdouble ∗ph, gdouble ∗ch)

    Compute the rotation angles (θ, ϕ, χ) between axes.
- gint **wbfmm_coefficients_H_rotation** (gdouble ∗H, gint N, gdouble th, gdouble ∗work)

    Compute rotation coefficients for angle θ.
- gint **wbfmm_expansion_laplace_cfft** (gint N, gdouble ∗x0, gdouble ∗xs, gdouble ∗q, gint nq, gdouble ∗cfft, gint cstr, gdouble ∗work)

    Generation of singular expansion coefficients for point source in Laplace problem.
- gint **wbfmm_expansion_laplace_evaluate** (gdouble ∗x0, gdouble ∗cfft, gint cstr, gint N, gint nq, gdouble ∗xf, gdouble ∗field, gdouble ∗work)

    Evaluate a singular expansion for Laplace problem.
- gint **wbfmm_expansion_laplace_evaluate_f** (gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gint nq, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

    Evaluate a singular expansion for Laplace problem.
- gint **wbfmm_expansion_laplace_cfft_f** (gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗q, gint nq, gfloat ∗cfft, gint cstr, gfloat ∗work)

    Generation of singular expansion coefficients for point source in Laplace problem.
- gint **wbfmm_coaxial_translate_SS_laplace** (gdouble ∗Co, gint cstro, gint No, gdouble ∗Ci, gint cstri, gint Ni, gint nq, gdouble t)

    Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.

- gint **wbfmm_coaxial_translate_SS_laplace_f** (gfloat ∗Co, gint cstro, gint No, gfloat ∗Ci, gint cstri, gint Ni, gint nq, gfloat t)

  Translate a singular expansion for the Laplace problem along the z axis to a singular expansion about a new centre.
- gint **wbfmm_child_parent_shift** (gdouble ∗Cp, gint Np, gdouble ∗Cc, gint Nc, gdouble ∗H03, gdouble ∗H47, gint Lh, gdouble ∗shift, gint Ls, gdouble ∗work)

  Upward shift of singular expansion from eight children to common parent.
- gint **wbfmm_parent_child_shift** (gdouble ∗Cc, gint Nc, gdouble ∗Cp, gint Np, gdouble ∗H03, gdouble ∗H47, gint Lh, gdouble ∗shift, gint Ls, gdouble ∗work)

  Downward shift of parent expansion to child box centres.
- gint **wbfmm_shift_angles_list4** (gint i, gint j, gint k, gdouble ∗th, gdouble ∗ph, gdouble ∗ch, gdouble ∗rs)

  Extract the rotation angles for boxes on interaction list 4.
- gint **wbfmm_shift_angle_table_init** (void)

  Initialize table of angles for shift operations.
- **wbfmm_shift_operators_t** ∗ **wbfmm_shift_operators_new** (guint L, gdouble ∗work)

  Allocate shift operators and initialize rotations.
- gint **wbfmm_upward_pass** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gdouble ∗work)

  Perform upward pass at one level of an octree.
- gint **wbfmm_downward_pass** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gdouble ∗work)

  Perform downward pass at one level of an octree.
- gint **wbfmm_tree_refine** (**wbfmm_tree_t** ∗t)

  Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.
- gint **wbfmm_tree_add_level** (**wbfmm_tree_t** ∗tree)
- gint **wbfmm_tree_add_points** (**wbfmm_tree_t** ∗t, gpointer pts, guint npts, gsize stride)

  Add points to an octree.
- guint64 **wbfmm_point_index_3d** (gdouble ∗x, gdouble ∗c, gdouble D)

  Find Morton index for point in a cubic domain.
- **wbfmm_tree_t** ∗ **wbfmm_tree_new** (gdouble ∗x, gdouble D, guint maxpoints)

  Allocate a new octree.
- gint **wbfmm_tree_coefficient_init** (**wbfmm_tree_t** ∗t, guint l, guint nr, guint ns)

  Initialize expansion coefficient data in an octree.
- gint **wbfmm_tree_leaf_expansions** (**wbfmm_tree_t** ∗t, gdouble k, gdouble ∗src, gint sstr, gdouble ∗normals, gint nstr, gdouble ∗dipoles, gint dstr, gboolean zero_expansions, gdouble ∗work)

  Generate leaf expansions for a tree.
- gint **wbfmm_box_location_from_index** (guint64 i, guint32 level, gdouble ∗x0, gdouble D, gdouble ∗x, gdouble ∗wb)

  Find the coordinates of a box from its Morton index.
- gint **wbfmm_shift_angles** (gdouble ∗xi, gdouble ∗xj, gdouble ∗th, gdouble ∗ph, gdouble ∗ch, gdouble ∗r)

  Compute angles and distance to shift expansion between two points.
- gint **wbfmm_tree_write_sources** (**wbfmm_tree_t** ∗t, gdouble ∗q, gint stride, FILE ∗f)

  Write a tree source list to file.
- gint **wbfmm_shift_coordinates_f** (gfloat ∗x, gfloat ∗y, gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗r)

  Find system of axes for coordinate shift.
- gint **wbfmm_legendre_recursion_array_f** (gfloat ∗∗Pnm1, gfloat ∗∗Pn, gint n, gfloat C, gfloat S)

  Perform recursion on normalized associated Legendre functions.
- gint **wbfmm_bessel_j_recursion_f** (gfloat ∗jnm1, gfloat ∗jn, gfloat x, gint n)

  Perform recursion on spherical Bessel function $j_n(x)$.
- gint **wbfmm_bessel_h_recursion_f** (gfloat ∗hnm1, gfloat ∗hn, gfloat x, gint n)

  Perform one step of spherical Hankel recursion.
- gint **wbfmm_bessel_j_init_f** (gfloat x, gfloat ∗j0, gfloat ∗j1)

  Initialize the spherical Bessel function recursion.

- gint **wbfmm_bessel_h_init_f** (gfloat x, gfloat ∗h0, gfloat ∗h1)

  Initialize spherical Hankel function recursion.
- gint **wbfmm_legendre_init_f** (gfloat C, gfloat S, gfloat ∗P0, gfloat ∗P10, gfloat ∗P11)

  Initialize normalized associated Legendre functions.
- gint **wbfmm_expansion_h_cfft_f** (gfloat k, gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗q, gfloat ∗cfft, gint cstr, gfloat ∗work)

  Generation of singular expansion coefficients for point source.
- gint **wbfmm_expansion_dipole_h_cfft_f** (gfloat k, gint N, gfloat ∗x0, gfloat ∗xs, gfloat ∗fx, gfloat ∗fy, gfloat ∗fz, gfloat ∗cfft, gint cstr, gfloat ∗work)

  Generation of singular expansion coefficients for point dipole source.
- gint **wbfmm_expansion_h_evaluate_f** (gfloat k, gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

  Evaluate a singular expansion.
- gint **wbfmm_expansion_j_evaluate_f** (gfloat k, gfloat ∗x0, gfloat ∗cfft, gint cstr, gint N, gfloat ∗xf, gfloat ∗field, gfloat ∗work)

  Evaluate a regular expansion.
- gint **wbfmm_total_dipole_field_f** (gfloat k, gfloat ∗xs, gint xstride, gfloat ∗src, gint sstride, gint nsrc, gfloat ∗xf, gfloat ∗field)

  Compute total field from dipole sources by direct evaluation.
- gint **wbfmm_coordinate_transform_f** (gfloat ∗x, gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗y)

  Transform coordinates to rotated axes.
- gint **wbfmm_coefficients_RR_coaxial_f** (gfloat ∗cfftRR, gint L, gfloat kr, gfloat ∗work)

  Generate coefficients for coaxial regular-to-regular translation.
- gint **wbfmm_coefficients_SR_coaxial_f** (gfloat ∗cfftSR, gint L, gfloat kr, gfloat ∗work)

  Generate coefficients for coaxial singular-to-regular translation.
- gint **wbfmm_rotation_angles_f** (gfloat ∗ix, gfloat ∗iy, gfloat ∗iz, gfloat ∗jx, gfloat ∗jy, gfloat ∗jz, gfloat ∗th, gfloat ∗ph, gfloat ∗ch)

  Compute the rotation angles (θ, ϕ, χ) between axes.
- gint **wbfmm_coefficients_H_rotation_f** (gfloat ∗H, gint N, gfloat th, gfloat ∗work)

  Compute rotation coefficients for angle θ.
- guint64 **wbfmm_point_index_3d_f** (gfloat ∗x, gfloat ∗c, gfloat D)

  Find Morton index for point in a cubic domain.
- **wbfmm_tree_t** ∗ **wbfmm_tree_new_f** (gfloat ∗x, gfloat D, guint maxpoints)

  Allocate a new octree.
- gint **wbfmm_tree_coefficient_init_f** (**wbfmm_tree_t** ∗t, guint l, guint nr, guint ns)

  Initialize expansion coefficient data in an octree.
- gint **wbfmm_tree_leaf_expansions_f** (**wbfmm_tree_t** ∗t, gfloat k, gfloat ∗src, gint sstr, gfloat ∗normals, gint nstr, gfloat ∗dipoles, gint dstr, gboolean zero_expansions, gfloat ∗work)

  Generate leaf expansions for a tree.
- gint **wbfmm_tree_refine_f** (**wbfmm_tree_t** ∗t)

  Refine an existing octree by adding a level and redistributing points attached to the tree to the boxes at the new level.
- gint **wbfmm_tree_add_points_f** (**wbfmm_tree_t** ∗t, gpointer pts, guint npts, gsize stride)

  Add points to an octree.
- gint **wbfmm_box_location_from_index_f** (guint64 i, guint32 level, gfloat ∗x0, gfloat D, gfloat ∗x, gfloat ∗wb)

  Find the coordinates of a box from its Morton index.
- gint **wbfmm_child_parent_shift_f** (gfloat ∗Cp, gint Np, gfloat ∗Cc, gint Nc, gfloat ∗H03, gfloat ∗H47, gint Lh, gfloat ∗shift, gint Ls, gfloat ∗work)

  Upward shift of singular expansion from eight children to common parent.
- gint **wbfmm_parent_child_shift_f** (gfloat ∗Cc, gint Nc, gfloat ∗Cp, gint Np, gfloat ∗H03, gfloat ∗H47, gint Lh, gfloat ∗shift, gint Ls, gfloat ∗work)

  Downward shift of parent expansion to child box centres.
- gint **wbfmm_shift_angles_list4_f** (gint i, gint j, gint k, gfloat ∗th, gfloat ∗ph, gfloat ∗ch, gfloat ∗rs)

Extract the rotation angles for boxes on interaction list 4.

- gint **wbfmm_shift_angles_f** (gfloat ∗xi, gfloat ∗xj, gfloat ∗th, gfloat ∗ph, gfloat ∗ch, gfloat ∗r)

  Compute angles and distance to shift expansion between two points.
- gint **wbfmm_tree_write_sources_f** (**wbfmm_tree_t** ∗t, gfloat ∗q, gint stride, FILE ∗f)

  Write a tree source list to file.
- gint **wbfmm_shift_angle_table_init_f** (void)

  Initialize table of angles for shift operations.
- **wbfmm_shift_operators_t** ∗ **wbfmm_shift_operators_new_f** (guint L, gfloat ∗work)

  Allocate shift operators and initialize rotations.
- gint **wbfmm_upward_pass_f** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gfloat ∗work)

  Perform upward pass at one level of an octree.
- gint **wbfmm_downward_pass_f** (**wbfmm_tree_t** ∗t, **wbfmm_shift_operators_t** ∗op, guint level, gfloat ∗work)

  Perform downward pass at one level of an octree.
- guint64 **wbfmm_box_index** (guint32 i, guint32 j, guint32 k)
- gint **wbfmm_box_location** (guint64 idx, guint32 ∗i, guint32 ∗j, guint32 ∗k)

7.2.1 Detailed Description

Header for Wide Band FMM library.

Author

Michael Carley `ensmjc@rpc-ensmjc.bath.ac.uk`

Date

Mon Jun 24 10:28:46 2019

Index