

WBFMM

Generated by Doxygen 1.8.9.1

Wed Nov 20 2019 12:13:40



# Contents

<b>1</b>	<b>WBFMM: A Wide Band Fast Multipole Method library</b>	<b>1</b>
1.1	Getting started . . . . .	1
1.2	What WBFMM does . . . . .	1
1.3	References . . . . .	1
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Data Structure Index</b>	<b>5</b>
3.1	Data Structures . . . . .	5
<b>4</b>	<b>File Index</b>	<b>7</b>
4.1	File List . . . . .	7
<b>5</b>	<b>Module Documentation</b>	<b>9</b>
5.1	Boxes and octrees . . . . .	9
5.1.1	Detailed Description . . . . .	10
5.1.2	Enumeration Type Documentation . . . . .	10
5.1.2.1	wbfmm_problem_t . . . . .	10
5.1.3	Function Documentation . . . . .	11
5.1.3.1	wbfmm_point_index_3d . . . . .	11
5.1.3.2	wbfmm_point_index_3d_f . . . . .	12
5.1.3.3	wbfmm_tree_add_level . . . . .	12
5.1.3.4	wbfmm_tree_add_points . . . . .	12
5.1.3.5	wbfmm_tree_add_points_f . . . . .	13
5.1.3.6	wbfmm_tree_box_field . . . . .	13
5.1.3.7	wbfmm_tree_box_field_f . . . . .	13
5.1.3.8	wbfmm_tree_box_local_field . . . . .	14
5.1.3.9	wbfmm_tree_box_local_field_f . . . . .	15
5.1.3.10	wbfmm_tree_coefficient_init . . . . .	15
5.1.3.11	wbfmm_tree_coefficient_init_f . . . . .	15
5.1.3.12	wbfmm_tree_laplace_box_local_field . . . . .	16
5.1.3.13	wbfmm_tree_laplace_box_local_field_f . . . . .	16

5.1.3.14	wbfmm_tree_laplace_leaf_expansions	17
5.1.3.15	wbfmm_tree_laplace_leaf_expansions_f	17
5.1.3.16	wbfmm_tree_leaf_expansions	18
5.1.3.17	wbfmm_tree_leaf_expansions_f	19
5.1.3.18	wbfmm_tree_new	19
5.1.3.19	wbfmm_tree_new_f	20
5.1.3.20	wbfmm_tree_refine	20
5.1.3.21	wbfmm_tree_refine_f	20
5.2	Shift operations	21
5.2.1	Detailed Description	21
5.2.2	Function Documentation	22
5.2.2.1	wbfmm_child_parent_shift	22
5.2.2.2	wbfmm_child_parent_shift_bw	22
5.2.2.3	wbfmm_child_parent_shift_bw_f	23
5.2.2.4	wbfmm_child_parent_shift_f	23
5.2.2.5	wbfmm_parent_child_shift	24
5.2.2.6	wbfmm_parent_child_shift_f	24
5.2.2.7	wbfmm_shift_angle_table_init	25
5.2.2.8	wbfmm_shift_angle_table_init_f	25
5.2.2.9	wbfmm_shift_angles_list4	25
5.2.2.10	wbfmm_shift_angles_list4_f	26
5.2.2.11	wbfmm_shift_operators_coaxial_SR_init	26
5.2.2.12	wbfmm_shift_operators_coaxial_SR_init	26
5.2.2.13	wbfmm_shift_operators_coaxial_SS_init	27
5.2.2.14	wbfmm_shift_operators_coaxial_SS_init	27
5.2.2.15	wbfmm_shift_operators_new	27
5.2.2.16	wbfmm_shift_operators_new_f	28
5.3	Generation and evaluation of expansions	29
5.3.1	Detailed Description	29
5.3.2	Function Documentation	29
5.3.2.1	wbfmm_expansion_dipole_h_cfft	29
5.3.2.2	wbfmm_expansion_dipole_h_cfft_f	30
5.3.2.3	wbfmm_expansion_h_cfft	30
5.3.2.4	wbfmm_expansion_h_cfft_f	31
5.3.2.5	wbfmm_expansion_h_evaluate	31
5.3.2.6	wbfmm_expansion_h_evaluate_f	31
5.3.2.7	wbfmm_expansion_j_evaluate	32
5.3.2.8	wbfmm_expansion_j_evaluate_f	32
5.4	Upward and downward passes	33
5.4.1	Detailed Description	33

5.4.2	Function Documentation . . . . .	33
5.4.2.1	wbfmm_downward_pass . . . . .	33
5.4.2.2	wbfmm_downward_pass_f . . . . .	34
5.4.2.3	wbfmm_laplace_downward_pass . . . . .	34
5.4.2.4	wbfmm_laplace_downward_pass_f . . . . .	34
5.4.2.5	wbfmm_laplace_upward_pass . . . . .	35
5.4.2.6	wbfmm_laplace_upward_pass_f . . . . .	35
5.4.2.7	wbfmm_upward_pass . . . . .	35
5.4.2.8	wbfmm_upward_pass_f . . . . .	36
5.5	Rotation coefficients and operations . . . . .	37
5.5.1	Detailed Description . . . . .	37
5.5.2	Function Documentation . . . . .	37
5.5.2.1	wbfmm_coefficients_H_rotation . . . . .	37
5.5.2.2	wbfmm_coefficients_H_rotation_f . . . . .	38
5.5.2.3	wbfmm_laplace_rotate_H . . . . .	38
5.5.2.4	wbfmm_laplace_rotate_H_f . . . . .	38
5.5.2.5	wbfmm_rotate_H . . . . .	39
5.5.2.6	wbfmm_rotate_H_f . . . . .	39
5.5.2.7	wbfmm_rotation_angles . . . . .	40
5.5.2.8	wbfmm_rotation_angles_f . . . . .	40
5.6	Translation operators . . . . .	42
5.6.1	Detailed Description . . . . .	42
5.6.2	Function Documentation . . . . .	42
5.6.2.1	wbfmm_coaxial_translate . . . . .	42
5.6.2.2	wbfmm_coaxial_translate_f . . . . .	43
5.6.2.3	wbfmm_coefficients_RR_coaxial . . . . .	43
5.6.2.4	wbfmm_coefficients_RR_coaxial_f . . . . .	43
5.6.2.5	wbfmm_coefficients_SR_coaxial . . . . .	44
5.6.2.6	wbfmm_coefficients_SR_coaxial_f . . . . .	44
5.7	Utility and convenience functions . . . . .	45
5.7.1	Detailed Description . . . . .	46
5.7.2	Function Documentation . . . . .	46
5.7.2.1	wbfmm_bessel_h_init . . . . .	46
5.7.2.2	wbfmm_bessel_h_init_f . . . . .	46
5.7.2.3	wbfmm_bessel_h_recursion . . . . .	47
5.7.2.4	wbfmm_bessel_h_recursion_f . . . . .	47
5.7.2.5	wbfmm_bessel_j_init . . . . .	47
5.7.2.6	wbfmm_bessel_j_init_f . . . . .	48
5.7.2.7	wbfmm_bessel_j_recursion . . . . .	48
5.7.2.8	wbfmm_bessel_j_recursion_f . . . . .	48

5.7.2.9	wbfmm_box_location_from_index . . . . .	49
5.7.2.10	wbfmm_box_location_from_index_f . . . . .	49
5.7.2.11	wbfmm_cartesian_to_spherical . . . . .	49
5.7.2.12	wbfmm_cartesian_to_spherical_f . . . . .	50
5.7.2.13	wbfmm_coordinate_transform . . . . .	50
5.7.2.14	wbfmm_coordinate_transform_f . . . . .	50
5.7.2.15	wbfmm_legendre_init . . . . .	51
5.7.2.16	wbfmm_legendre_init_f . . . . .	51
5.7.2.17	wbfmm_legendre_recursion_array . . . . .	51
5.7.2.18	wbfmm_legendre_recursion_array_f . . . . .	52
5.7.2.19	wbfmm_points_origin_width . . . . .	52
5.7.2.20	wbfmm_points_origin_width_f . . . . .	52
5.7.2.21	wbfmm_shift_angles . . . . .	53
5.7.2.22	wbfmm_shift_angles_f . . . . .	53
5.7.2.23	wbfmm_shift_coordinates . . . . .	53
5.7.2.24	wbfmm_shift_coordinates_f . . . . .	54
5.7.2.25	wbfmm_total_dipole_field . . . . .	54
5.7.2.26	wbfmm_total_dipole_field_f . . . . .	54
5.7.2.27	wbfmm_total_field . . . . .	55
5.7.2.28	wbfmm_total_field_f . . . . .	55
5.7.2.29	wbfmm_tree_box_centre . . . . .	56
5.7.2.30	wbfmm_tree_box_centre_f . . . . .	56
5.7.2.31	wbfmm_tree_write_sources . . . . .	56
5.7.2.32	wbfmm_tree_write_sources_f . . . . .	57
5.8	Evaluation of the Laplace potential . . . . .	58
5.8.1	Detailed Description . . . . .	59
5.8.2	Function Documentation . . . . .	59
5.8.2.1	wbfmm_box_fields_laplace . . . . .	59
5.8.2.2	wbfmm_box_fields_laplace_f . . . . .	60
5.8.2.3	wbfmm_expansion_laplace_evaluate . . . . .	60
5.8.2.4	wbfmm_expansion_laplace_evaluate_f . . . . .	60
5.8.2.5	wbfmm_laplace_child_parent_shift . . . . .	61
5.8.2.6	wbfmm_laplace_child_parent_shift_f . . . . .	61
5.8.2.7	wbfmm_laplace_coaxial_translate_init . . . . .	62
5.8.2.8	wbfmm_laplace_coaxial_translate_init_f . . . . .	62
5.8.2.9	wbfmm_laplace_coaxial_translate_RR . . . . .	62
5.8.2.10	wbfmm_laplace_coaxial_translate_RR_f . . . . .	63
5.8.2.11	wbfmm_laplace_coaxial_translate_SR . . . . .	63
5.8.2.12	wbfmm_laplace_coaxial_translate_SR_f . . . . .	64
5.8.2.13	wbfmm_laplace_coaxial_translate_SS . . . . .	64

5.8.2.14	wbfmm_laplace_coaxial_translate_SS_f . . . . .	65
5.8.2.15	wbfmm_laplace_expansion_apply . . . . .	65
5.8.2.16	wbfmm_laplace_expansion_apply_f . . . . .	65
5.8.2.17	wbfmm_laplace_expansion_cfft . . . . .	66
5.8.2.18	wbfmm_laplace_expansion_cfft_f . . . . .	66
5.8.2.19	wbfmm_laplace_expansion_local_evaluate . . . . .	66
5.8.2.20	wbfmm_laplace_expansion_local_evaluate_f . . . . .	67
5.8.2.21	wbfmm_laplace_field . . . . .	67
5.8.2.22	wbfmm_laplace_field_coefficients . . . . .	67
5.8.2.23	wbfmm_laplace_field_coefficients_f . . . . .	67
5.8.2.24	wbfmm_laplace_field_f . . . . .	67
5.8.2.25	wbfmm_laplace_local_coefficients . . . . .	68
5.8.2.26	wbfmm_laplace_local_coefficients_f . . . . .	68
5.8.2.27	wbfmm_laplace_parent_child_shift . . . . .	68
5.8.2.28	wbfmm_laplace_parent_child_shift_f . . . . .	69
5.9	Indexing and lookup operations . . . . .	70
5.9.1	Detailed Description . . . . .	70
5.9.2	Function Documentation . . . . .	70
5.9.2.1	wbfmm_box_index . . . . .	70
5.9.2.2	wbfmm_box_location . . . . .	70
<b>6</b>	<b>Data Structure Documentation</b>	<b>71</b>
6.1	wbfmm_box_t Struct Reference . . . . .	71
6.1.1	Detailed Description . . . . .	71
6.1.2	Field Documentation . . . . .	71
6.1.2.1	i . . . . .	71
6.1.2.2	mpr . . . . .	71
6.1.2.3	mps . . . . .	71
6.1.2.4	n . . . . .	71
6.2	wbfmm_shift_operators_t Struct Reference . . . . .	71
6.2.1	Detailed Description . . . . .	72
6.2.2	Field Documentation . . . . .	72
6.2.2.1	L . . . . .	72
6.2.2.2	nerot . . . . .	72
6.2.2.3	nlevels . . . . .	72
6.2.2.4	rotations . . . . .	72
6.2.2.5	size . . . . .	72
6.2.2.6	SR . . . . .	72
6.2.2.7	SS . . . . .	72
6.3	wbfmm_target_list_t Struct Reference . . . . .	72

6.3.1	Detailed Description . . . . .	73
6.3.2	Field Documentation . . . . .	73
6.3.2.1	boxes . . . . .	73
6.3.2.2	cfft . . . . .	73
6.3.2.3	csrc . . . . .	73
6.3.2.4	grad . . . . .	73
6.3.2.5	ibox . . . . .	73
6.3.2.6	ics . . . . .	73
6.3.2.7	ip . . . . .	73
6.3.2.8	isrc . . . . .	74
6.3.2.9	maxpoints . . . . .	74
6.3.2.10	nc . . . . .	74
6.3.2.11	npoints . . . . .	74
6.3.2.12	points . . . . .	74
6.3.2.13	pstr . . . . .	74
6.3.2.14	size . . . . .	74
6.3.2.15	t . . . . .	74
6.4	wbfmm_tree_t Struct Reference . . . . .	74
6.4.1	Detailed Description . . . . .	75
6.4.2	Field Documentation . . . . .	75
6.4.2.1	boxes . . . . .	75
6.4.2.2	D . . . . .	75
6.4.2.3	depth . . . . .	75
6.4.2.4	ip . . . . .	75
6.4.2.5	maxpoints . . . . .	75
6.4.2.6	mpr . . . . .	75
6.4.2.7	mps . . . . .	75
6.4.2.8	npoints . . . . .	75
6.4.2.9	nq . . . . .	75
6.4.2.10	order_r . . . . .	75
6.4.2.11	order_s . . . . .	76
6.4.2.12	points . . . . .	76
6.4.2.13	pstr . . . . .	76
6.4.2.14	size . . . . .	76
6.4.2.15	x . . . . .	76
<b>7</b>	<b>File Documentation</b>	<b>77</b>
7.1	tree.c File Reference . . . . .	77
7.1.1	Detailed Description . . . . .	77
7.2	wbfmm.h File Reference . . . . .	77



7.2.1 Detailed Description . . . . .	83
<b>Index</b>	<b>85</b>



# 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.umi.acs.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.umi.acs.umd.edu/~ramani/pubs/comparisontranslationmethods\\_041205.pdf](http://www.umi.acs.umd.edu/~ramani/pubs/comparisontranslationmethods_041205.pdf)



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Boxes and octrees . . . . .	9
Shift operations . . . . .	21
Generation and evaluation of expansions . . . . .	29
Upward and downward passes . . . . .	33
Rotation coefficients and operations . . . . .	37
Translation operators . . . . .	42
Utility and convenience functions . . . . .	45
Evaluation of the Laplace potential . . . . .	58
Indexing and lookup operations . . . . .	70



## Chapter 3

# Data Structure Index

### 3.1 Data Structures

Here are the data structures with brief descriptions:

<b>wbfmm_box_t</b> . . . . .	71
<b>wbfmm_shift_operators_t</b> . . . . .	71
<b>wbfmm_target_list_t</b> . . . . .	72
<b>wbfmm_tree_t</b> . . . . .	74





## Chapter 4

# File Index

### 4.1 File List

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

<b>tree.c</b> . . . . .	77
<b>wbfmm.h</b>	
Header for Wide Band FMM library . . . . .	77



# 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\_target\_list\_t**
- struct **wbfmm\_shift\_operators\_t**

#### Enumerations

- enum **wbfmm\_problem\_t** { **WBFMM\_PROBLEM\_LAPLACE** = 1, **WBFMM\_PROBLEM\_HELMHOLTZ** = 2 }

#### Functions

- gint **wbfmm\_tree\_add\_points** (**wbfmm\_tree\_t** \*t, gpointer pts, guint npts, gsize pstr)  
*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.*
- gint **wbfmm\_tree\_box\_field** (**wbfmm\_tree\_t** \*t, guint level, guint b, gdouble k, gdouble \*x, gdouble \*f, gdouble \*work)  
*Evaluate singular expansion about a box centre.*
- gint **wbfmm\_tree\_box\_local\_field** (**wbfmm\_tree\_t** \*t, guint level, guint b, gdouble k, gdouble \*x, gdouble \*f, gdouble \*src, gint sstr, gboolean eval\_neighbours, gdouble \*work)  
*Evaluate local field from regular expansion in box.*
- 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\_laplace\_box\_local\_field** (wbfmm\_tree\_t \*t, quint level, quint b, gdouble \*x, gdouble \*f, gdouble \*src, gint sstr, gdouble \*normals, gint nstr, gdouble \*d, gint dstr, gboolean eval\_neighbours, gdouble \*work)

*Evaluate local Laplace field from regular expansion in box.*

- gint **wbfmm\_tree\_laplace\_leaf\_expansions** (wbfmm\_tree\_t \*t, gdouble \*src, gint sstr, gdouble \*normals, gint nstr, gdouble \*dipoles, gint dstr, gboolean zero\_expansions, gdouble \*work)

*Generate leaf expansions for a tree in the Laplace problem.*

- gint **wbfmm\_tree\_add\_points\_f** (wbfmm\_tree\_t \*t, gpointer pts, quint npts, gsize pstr)

*Add points to an octree.*

- wbfmm\_tree\_t \* **wbfmm\_tree\_new\_f** (gfloat \*x, gfloat D, quint 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.*

- gint **wbfmm\_tree\_box\_field\_f** (wbfmm\_tree\_t \*t, quint level, quint b, gfloat k, gfloat \*x, gfloat \*f, gfloat \*work)

*Evaluate singular expansion about a box centre.*

- gint **wbfmm\_tree\_box\_local\_field\_f** (wbfmm\_tree\_t \*t, quint level, quint b, gfloat k, gfloat \*x, gfloat \*f, gfloat \*src, gint sstr, gboolean eval\_neighbours, gfloat \*work)

*Evaluate local field from regular expansion in box.*

- 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, quint l, quint nr, quint 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.*

- gint **wbfmm\_tree\_laplace\_box\_local\_field\_f** (wbfmm\_tree\_t \*t, quint level, quint b, gfloat \*x, gfloat \*f, gfloat \*src, gint sstr, gfloat \*normals, gint nstr, gfloat \*d, gint dstr, gboolean eval\_neighbours, gfloat \*work)

*Evaluate local Laplace field from regular expansion in box.*

- gint **wbfmm\_tree\_laplace\_leaf\_expansions\_f** (wbfmm\_tree\_t \*t, gfloat \*src, gint sstr, gfloat \*normals, gint nstr, gfloat \*dipoles, gint dstr, gboolean zero\_expansions, gfloat \*work)

*Generate leaf expansions for a tree in the Laplace problem.*

- gint **wbfmm\_tree\_add\_level** (wbfmm\_tree\_t \*t)

### 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. 74)

Enumerator

**WBFMM\_PROBLEM\_LAPLACE** Laplace equation

**WBFMM\_PROBLEM\_HELMHOLTZ** Helmholtz equation

### 5.1.3 Function Documentation

5.1.3.1 `uint64 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. 71)

**Parameters**

$t$	an existing <b>wbfmm_tree_t</b> (p. 74)
-----	---

**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 <b>wbfmm_tree_t</b> (p. 74);
$pts$	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

<i>t</i>	an existing <b>wbfmm_tree_t</b> (p. 74);
<i>pts</i>	an array containing point coordinates;
<i>npts</i>	the number of points in <i>pts</i> ;
<i>pstr</i>	stride between points in bytes.

#### Returns

0 on success.

### 5.1.3.6 `gint wbfmm_tree_box_field ( wbfmm_tree_t * t, guint level, guint b, gdouble k, gdouble * x, gdouble * f, gdouble * work )`

Evaluate singular expansion about a box centre.

#### Parameters

<i>t</i>	octree for problem;
<i>level</i>	level in <i>t</i> ;
<i>b</i>	index of box at level <i>level</i> of <i>t</i> ;
<i>k</i>	wavenumber;
<i>x</i>	field evaluation point;
<i>f</i>	on output field at <i>x</i> (not zeroed before evaluation);
<i>work</i>	workspace

#### Returns

0 on success

### 5.1.3.7 `gint wbfmm_tree_box_field_f ( wbfmm_tree_t * t, guint level, guint b, gfloat k, gfloat * x, gfloat * f, gfloat * work )`

Evaluate singular expansion about a box centre.

#### Parameters

<i>t</i>	octree for problem;
<i>level</i>	level in <i>t</i> ;
<i>b</i>	index of box at level <i>level</i> of <i>t</i> ;
<i>k</i>	wavenumber;
<i>x</i>	field evaluation point;
<i>f</i>	on output field at <i>x</i> (not zeroed before evaluation);
<i>work</i>	workspace

#### Returns

0 on success

5.1.3.8 `gint wbfmm_tree_box_local_field ( wbfmm_tree_t * t, guint level, guint b, gdouble k, gdouble * x, gdouble * f, gdouble * src, gint sstr, gboolean eval_neighbours, gdouble * work )`

Evaluate local field from regular expansion in box.



## Parameters

<i>t</i>	octree for domain;
<i>level</i>	level of <i>t</i> ;
<i>b</i>	box index at level <i>level</i> of <i>t</i> ;
<i>k</i>	wavenumber;
<i>x</i>	location of evaluation point;
<i>f</i>	on output, field value (not zeroed before evaluation);
<i>src</i>	source strengths;
<i>sstr</i>	stride of data in <i>src</i> ;
<i>eval_neighbours</i>	if TRUE compute contributions from sources in box <i>b</i> and neighbours;
<i>work</i>	workspace.

## Returns

0 on success

5.1.3.9 `gint wbfmm_tree_box_local_field_f ( wbfmm_tree_t * t, quint level, quint b, gfloat k, gfloat * x, gfloat * f, gfloat * src, gint sstr, gboolean eval_neighbours, gfloat * work )`

Evaluate local field from regular expansion in box.

## Parameters

<i>t</i>	octree for domain;
<i>level</i>	level of <i>t</i> ;
<i>b</i>	box index at level <i>level</i> of <i>t</i> ;
<i>k</i>	wavenumber;
<i>x</i>	location of evaluation point;
<i>f</i>	on output, field value (not zeroed before evaluation);
<i>src</i>	source strengths;
<i>sstr</i>	stride of data in <i>src</i> ;
<i>eval_neighbours</i>	if TRUE compute contributions from sources in box <i>b</i> and neighbours;
<i>work</i>	workspace.

## Returns

0 on success

5.1.3.10 `gint wbfmm_tree_coefficient_init ( wbfmm_tree_t * t, quint l, quint nr, quint ns )`

Initialize expansion coefficient data in an octree.

## Parameters

<i>t</i>	octree for problem;
<i>l</i>	level to initialize data for;
<i>nr</i>	order of regular expansions at level <i>l</i> ;
<i>ns</i>	order of singular expansions at level <i>l</i> .

## Returns

0 on success

5.1.3.11 `gint wbfmm_tree_coefficient_init_f ( wbfmm_tree_t * t, quint l, quint nr, quint ns )`

Initialize expansion coefficient data in an octree.

## Parameters

<i>t</i>	octree for problem;
<i>l</i>	level to initialize data for;
<i>nr</i>	order of regular expansions at level <i>l</i> ;
<i>ns</i>	order of singular expansions at level <i>l</i> .

## Returns

0 on success

5.1.3.12 `gint wbfmm_tree_laplace_box_local_field ( wbfmm_tree_t * t, guint level, guint b, gdouble * x, gdouble * f, gdouble * src, gint sstr, gdouble * normals, gint nstr, gdouble * d, gint dstr, gboolean eval_neighbours, gdouble * work )`

Evaluate local Laplace field from regular expansion in box.

## Parameters

<i>t</i>	octree for domain;
<i>level</i>	level of <i>t</i> ;
<i>b</i>	box index at level <i>level</i> of <i>t</i> ;
<i>x</i>	location of evaluation point;
<i>f</i>	on output, field value (not zeroed before evaluation);
<i>src</i>	source strengths;
<i>sstr</i>	stride of data in <i>src</i> ;
<i>normals</i>	normals for dipole sources;
<i>nstr</i>	stride in <i>normals</i> ;
<i>d</i>	dipole source strengths, or dipole vectors;
<i>dstr</i>	stride in <i>d</i> ;
<i>eval_neighbours</i>	if TRUE compute contributions from sources in box <i>b</i> and neighbours;
<i>work</i>	workspace.

## Returns

0 on success

5.1.3.13 `gint wbfmm_tree_laplace_box_local_field_f ( wbfmm_tree_t * t, guint level, guint b, gfloat * x, gfloat * f, gfloat * src, gint sstr, gfloat * normals, gint nstr, gfloat * d, gint dstr, gboolean eval_neighbours, gfloat * work )`

Evaluate local Laplace field from regular expansion in box.

## Parameters

<i>t</i>	octree for domain;
<i>level</i>	level of <i>t</i> ;
<i>b</i>	box index at level <i>level</i> of <i>t</i> ;
<i>x</i>	location of evaluation point;
<i>f</i>	on output, field value (not zeroed before evaluation);
<i>src</i>	source strengths;
<i>sstr</i>	stride of data in <i>src</i> ;

<i>normals</i>	normals for dipole sources;
<i>nstr</i>	stride in <i>normals</i> ;
<i>d</i>	dipole source strengths, or dipole vectors;
<i>dstr</i>	stride in <i>d</i> ;
<i>eval_neighbours</i>	if TRUE compute contributions from sources in box <i>b</i> and neighbours;
<i>work</i>	workspace.

## Returns

0 on success

**5.1.3.14** `gint wbfmm_tree_laplace_leaf_expansions ( wbfmm_tree_t * t, gdouble * src, gint sstr, gdouble * normals, gint nstr, gdouble * dipoles, gint dstr, gboolean zero_expansions, gdouble * work )`

Generate leaf expansions for a tree in the Laplace problem.

Generate leaf expansions for a tree for the Laplace problem 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:

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

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

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

## Returns

0 on success

**5.1.3.15** `gint wbfmm_tree_laplace_leaf_expansions_f ( wbfmm_tree_t * t, gfloat * src, gint sstr, gfloat * normals, gint nstr, gfloat * dipoles, gint dstr, gboolean zero_expansions, gfloat * work )`

Generate leaf expansions for a tree in the Laplace problem.

Generate leaf expansions for a tree for the Laplace problem 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. 13)(...) 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:

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

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

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

#### Returns

0 on success

**5.1.3.16** `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:

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

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

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

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

**Returns**

0 on success

**5.1.3.17** `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. 13)(...) 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:

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

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**

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

**Returns**

0 on success

**5.1.3.18** `wbfmm_tree_t* wbfmm_tree_new ( gdouble * x, gdouble D, quint maxpoints )`

Allocate a new octree.

**Parameters**

<i>x</i>	location of origin of tree;
<i>D</i>	width of domain;

<i>maxpoints</i>	maximum number of source points in tree.
------------------	--

**Returns**

pointer to newly allocated tree.

**5.1.3.19 wbfmm\_tree\_t\* wbfmm\_tree\_new\_f ( gfloat \* x, gfloat D, quint maxpoints )**

Allocate a new octree.

**Parameters**

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

**Returns**

pointer to newly allocated tree.

**5.1.3.20 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**

<i>t</i>	an existing <b>wbfmm_tree_t</b> (p. 74).
----------	--

**Returns**

0 on success.

**5.1.3.21 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**

<i>t</i>	an existing <b>wbfmm_tree_t</b> (p. 74).
----------	--

**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 \*trans, gint Ls, gdouble \*work)  
*Upward shift of singular expansion from eight children to common parent.*
- gint **wbfmm\_child\_parent\_shift\_bw** (gdouble \*Cp, gint Np, gdouble \*Cc, gint Nc, gdouble \*H03, gint Lh, gdouble \*transf, gdouble \*transb, gint Ls, gdouble \*work)  
*Upward shift of singular expansion from eight children to common parent, using backward translations.*
- 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.*
- 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\_shift\_operators\_coaxial\_SR\_init** (**wbfmm\_shift\_operators\_t** \*w, gdouble D, guint level, guint L, gdouble k, gdouble \*work)  
*Initialize singular-to-regular translation operators.*
- gint **wbfmm\_shift\_operators\_coaxial\_SS\_init** (**wbfmm\_shift\_operators\_t** \*w, gdouble D, guint level, guint L, gdouble k, gdouble \*work)  
*Initialize singular-to-singular (regular-to-regular) translation operators.*
- 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.*
- gint **wbfmm\_child\_parent\_shift\_bw\_f** (gfloat \*Cp, gint Np, gfloat \*Cc, gint Nc, gfloat \*H03, gint Lh, gfloat \*transf, gfloat \*transb, gint Ls, gfloat \*work)  
*Upward shift of singular expansion from eight children to common parent, using backward translations.*
- 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.*
- 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.*
- gint **wbfmm\_shift\_operators\_coaxial\_SR\_init** (**wbfmm\_shift\_operators\_t** \*w, gfloat D, guint level, guint L, gfloat k, gfloat \*work)  
*Initialize singular-to-regular translation operators.*
- gint **wbfmm\_shift\_operators\_coaxial\_SS\_init** (**wbfmm\_shift\_operators\_t** \*w, gfloat D, guint level, guint L, gfloat k, gfloat \*work)  
*Initialize singular-to-singular (regular-to-regular) translation operators.*

### 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

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

### Returns

0 on success

**5.2.2.2** `gint wbfmm_child_parent_shift_bw ( gdouble * Cp, gint Np, gdouble * Cc, gint Nc, gdouble * H03, gint Lh, gdouble * transf, gdouble * transb, gint Ls, gdouble * work )`

Upward shift of singular expansion from eight children to common parent, using backward translations.

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. The method is the same as for **wbfmm\_child\_parent\_shift** (p. 22)(...), except that the child boxes with Morton indices 4-7 are rotated in the same sense as the diagonally opposite child boxes 0-3 and a reverse (negative distance) coaxial translation is used to combine them with the lower child box data with the same rotation. The reverse rotation is then applied to the summed data meaning that only four reverse rotations rather than eight are required to transfer the data to the parent box orientation.

### Parameters

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>Lh</i>	maximum order of rotation coefficients;
<i>transf</i>	forward ( $+kr$ ) coaxial translation operator for distance between child and parent box centres;
<i>transb</i>	backward ( $-kr$ ) coaxial translation operator for distance between child and parent box centres;
<i>Ls</i>	order of <i>trans</i> ;



<i>work</i>	workspace
-------------	-----------

**Returns**

0 on success

**5.2.2.3** `gint wbfmm_child_parent_shift_bw_f ( gfloat * Cp, gint Np, gfloat * Cc, gint Nc, gfloat * H03, gint Lh, gfloat * transf, gfloat * transb, gint Ls, gfloat * work )`

Upward shift of singular expansion from eight children to common parent, using backward translations.

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. The method is the same as for **wbfmm\_child\_parent\_shift\_f** (p. 23)(...), except that the child boxes with Morton indices 4-7 are rotated in the same sense as the diagonally opposite child boxes 0-3 and a reverse (negative distance) coaxial translation is used to combine them with the lower child box data with the same rotation. The reverse rotation is then applied to the summed data meaning that only four reverse rotations rather than eight are required to transfer the data to the parent box orientation.

**Parameters**

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>Lh</i>	maximum order of rotation coefficients;
<i>transf</i>	forward ( $+kr$ ) coaxial translation operator for distance between child and parent box centres;
<i>transb</i>	backward ( $-kr$ ) coaxial translation operator for distance between child and parent box centres;
<i>Ls</i>	order of <i>trans</i> ;
<i>work</i>	workspace

**Returns**

0 on success

**5.2.2.4** `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**

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);

<i>H47</i>	rotation coefficients for 'upper' children (Morton index 4-7);
<i>Lh</i>	maximum order of rotation coefficients;
<i>trans</i>	coaxial translation operator for distance between child and parent box centres;
<i>Ls</i>	order of <i>trans</i> ;
<i>work</i>	workspace

#### Returns

0 on success

**5.2.2.5** `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. 22)(...). Note that the rotation matrices for this function are switched relative to the rotations of the same name in **wbfmm\_child\_parent\_shift** (p. 22)(...), 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

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

#### Returns

0 on success

**5.2.2.6** `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. 23)(...). 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. 23)(...), 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

<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;

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

**Returns**

0 on success

**5.2.2.7 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. 27)(...), in order to initialize the look-up table of orientations between boxes in interaction lists.

**Returns**

0 on success

**5.2.2.8 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. 28)(...), in order to initialize the look-up table of orientations between boxes in interaction lists.

**Returns**

0 on success

**5.2.2.9 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. 25)(...)

**Parameters**

<i>i</i>	integer $x$ coordinate of box on interaction list;
<i>j</i>	integer $y$ coordinate of box on interaction list;
<i>k</i>	integer $z$ coordinate of box on interaction list;
<i>th</i>	$\theta$ for rotation between boxes;
<i>ph</i>	$\phi$ for rotation between boxes;
<i>ch</i>	$\chi$ for rotation between boxes;

<i>rs</i>	scaling factor for distance between box centres, distance is <i>rs</i> multiplied by box width.
-----------	---

## Returns

0 on success

#### 5.2.2.10 `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. 25)(...)

## Parameters

<i>i</i>	integer $x$ coordinate of box on interaction list;
<i>j</i>	integer $y$ coordinate of box on interaction list;
<i>k</i>	integer $z$ coordinate of box on interaction list;
<i>th</i>	$\theta$ for rotation between boxes;
<i>ph</i>	$\phi$ for rotation between boxes;
<i>ch</i>	$\chi$ for rotation between boxes;
<i>rs</i>	scaling factor for distance between box centres, distance is <i>rs</i> multiplied by box width.

## Returns

0 on success

#### 5.2.2.11 `gint wbfmm_shift_operators_coaxial_SR_init ( wbfmm_shift_operators_t * w, gdouble D, quint level, quint L, gdouble k, gdouble * work )`

Initialize singular-to-regular translation operators.

## Parameters

<i>w</i>	a <code>wbfmm_shift_operators_t</code> (p. 71) allocated with <code>wbfmm_shift_operators_new(...)</code> ;
<i>D</i>	width of the problem domain;
<i>level</i>	level for which to generate translations;
<i>L</i>	order of translations;
<i>k</i>	wavenumber;
<i>work</i>	workspace

## Returns

0 on success

#### 5.2.2.12 `gint wbfmm_shift_operators_coaxial_SR_init ( wbfmm_shift_operators_t * w, gfloat D, quint level, quint L, gfloat k, gfloat * work )`

Initialize singular-to-regular translation operators.

## Parameters

<i>w</i>	a <b>wbfmm_shift_operators_t</b> (p. 71) allocated with <code>wbfmm_shift_operators_new_f(...)</code> ;
<i>D</i>	width of the problem domain;
<i>level</i>	level for which to generate translations;
<i>L</i>	order of translations;
<i>k</i>	wavenumber;
<i>work</i>	workspace

**Returns**

0 on success

**5.2.2.13** `gint wbfmm_shift_operators_coaxial_SS_init ( wbfmm_shift_operators_t * w, gdouble D, guint level, guint L, gdouble k, gdouble * work )`

Initialize singular-to-singular (regular-to-regular) translation operators.

**Parameters**

<i>w</i>	a <b>wbfmm_shift_operators_t</b> (p. 71) allocated with <code>wbfmm_shift_operators_new(...)</code> ;
<i>D</i>	width of the problem domain;
<i>level</i>	level for which to generate translations;
<i>L</i>	order of translations;
<i>k</i>	wavenumber;
<i>work</i>	workspace

**Returns**

0 on success

**5.2.2.14** `gint wbfmm_shift_operators_coaxial_SS_init ( wbfmm_shift_operators_t * w, gfloat D, guint level, guint L, gfloat k, gfloat * work )`

Initialize singular-to-singular (regular-to-regular) translation operators.

**Parameters**

<i>w</i>	a <b>wbfmm_shift_operators_t</b> (p. 71) allocated with <code>wbfmm_shift_operators_new_f(...)</code> ;
<i>D</i>	width of the problem domain;
<i>level</i>	level for which to generate translations;
<i>L</i>	order of translations;
<i>k</i>	wavenumber;
<i>work</i>	workspace

**Returns**

0 on success

**5.2.2.15** `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. 71) of given maximum order and initialize the rotation coefficients needed for same-level interaction calculations and upward and downward passes.

## Parameters

<i>L</i>	maximum order of expansions;
<i>work</i>	workspace.

## Returns

0 on success

**5.2.2.16 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. 71) of given maximum order and initialize the rotation coefficients needed for same-level interaction calculations and upward and downward passes.

## Parameters

<i>L</i>	maximum order of expansions;
<i>work</i>	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.*
- gfloat **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.*
- gfloat **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.*
- gfloat **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.*
- gfloat **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. 58)). 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

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

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

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



<i>cstr</i>	stride in <i>cfft</i> , in number of complex elements;
<i>work</i>	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**

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

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

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

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

<i>k</i>	wavenumber;
<i>x0</i>	centre of expansion;
<i>cfft</i>	expansion coefficients;
<i>cstr</i>	stride in <i>cfft</i> , in number of complex elements;
<i>N</i>	order of expansion;
<i>xf</i>	field point;
<i>field</i>	incremented with computed field;
<i>work</i>	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\_laplace\_downward\_pass** (wbfmm\_tree\_t \*t, wbfmm\_shift\_operators\_t \*op, guint level, gdouble \*work)  
*Perform downward pass at one level of an octree for the Laplace problem.*
- gint **wbfmm\_laplace\_upward\_pass** (wbfmm\_tree\_t \*t, wbfmm\_shift\_operators\_t \*op, guint level, gdouble \*work)  
*Perform upward pass at one level of an octree for the Laplace problem.*
- 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.*
- gint **wbfmm\_laplace\_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 for the Laplace problem.*
- gint **wbfmm\_laplace\_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 for the Laplace problem.*

### 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

<i>t</i>	an initialized octree which has had the upward pass performed;
<i>op</i>	shift operators allocated for <i>t</i> ;
<i>level</i>	level at which to perform downward pass;

<i>work</i>	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**

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

**Returns**

0 on success

**5.4.2.3** `gint wbfmm_laplace_downward_pass( wbfmm_tree_t * t, wbfmm_shift_operators_t * op, guint level, gdouble * work )`

Perform downward pass at one level of an octree for the Laplace problem.

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**

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

**Returns**

0 on success

**5.4.2.4** `gint wbfmm_laplace_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 for the Laplace problem.

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

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

## Returns

0 on success

5.4.2.5 `gint wbfmm_laplace_upward_pass ( wbfmm_tree_t * t, wbfmm_shift_operators_t * op, guint level, gdouble * work )`

Perform upward pass at one level of an octree for the Laplace problem.

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

<i>t</i>	an initialized octree;
<i>op</i>	shift operators allocated for <i>t</i> ;
<i>level</i>	level at which to perform upward pass;
<i>work</i>	workspace

## Returns

0 on success

5.4.2.6 `gint wbfmm_laplace_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 for the Laplace problem.

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

<i>t</i>	an initialized octree;
<i>op</i>	shift operators allocated for <i>t</i> ;
<i>level</i>	level at which to perform upward pass;
<i>work</i>	workspace

## Returns

0 on success

5.4.2.7 `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**

<i>t</i>	an initialized octree;
<i>op</i>	shift operators allocated for <i>t</i> ;
<i>level</i>	level at which to perform upward pass;
<i>work</i>	workspace

**Returns**

0 on success

5.4.2.8 `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**

<i>t</i>	an initialized octree;
<i>op</i>	shift operators allocated for <i>t</i> ;
<i>level</i>	level at which to perform upward pass;
<i>work</i>	workspace

**Returns**

0 on success

## 5.5 Rotation coefficients and operations

Computation and application of rotation operators.

### 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  $(\theta, \phi, \chi)$  between axes.*
- gint **wbfmm\_coefficients\_H\_rotation** (gdouble \*H, gint N, gdouble th, gdouble \*work)  
*Compute rotation coefficients for angle  $\theta$ .*
- gint **wbfmm\_rotate\_H** (gdouble \*Co, gint cstro, gdouble \*Ci, gint cstri, gint N, gdouble \*H, gdouble ph, gdouble ch)  
*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients.*
- gint **wbfmm\_laplace\_rotate\_H** (gdouble \*Co, gint cstro, gdouble \*Ci, gint cstri, gint N, gint nq, gdouble \*H, gdouble ph, gdouble ch)  
*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.*
- 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  $(\theta, \phi, \chi)$  between axes.*
- gint **wbfmm\_coefficients\_H\_rotation\_f** (gfloat \*H, gint N, gfloat th, gfloat \*work)  
*Compute rotation coefficients for angle  $\theta$ .*
- gint **wbfmm\_rotate\_H\_f** (gfloat \*Co, gint cstro, gfloat \*Ci, gint cstri, gint N, gfloat \*H, gfloat ph, gfloat ch)  
*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients.*
- gint **wbfmm\_laplace\_rotate\_H\_f** (gfloat \*Co, gint cstro, gfloat \*Ci, gint cstri, gint N, gint nq, gfloat \*H, gfloat ph, gfloat ch)  
*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.*

### 5.5.1 Detailed Description

Computation and application of rotation operators.

Recursive computation of rotation coefficients using the methods of Gumerov and Duraiswami, <http://dx.doi.org/10.1137/S1064827501399705>

### 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  $\theta$ .

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 $\theta$ , from <b>wbfmm_rotation_angles</b> (p. 40)(...);
$work$	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  $\theta$ .

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

<i>H</i>	on output rotation coefficients;
<i>N</i>	maximum order of coefficients to compute;
<i>th</i>	rotation angle $\theta$ , from <b>wbfmm_rotation_angles_f</b> (p. 40)(...);
<i>work</i>	workspace

#### Returns

0 on success

### 5.5.2.3 `gint wbfmm_laplace_rotate_H ( gdouble * Co, gint cstro, gdouble * Ci, gint cstri, gint N, gint nq, gdouble * H, gdouble ph, gdouble ch )`

Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.

Given the rotation coefficients  $H$  for angle  $\theta$  from **wbfmm\_coefficients\_H\_rotation** (p. 37)(...), rotate input coefficients to new system of axes, using  $H$  and angles  $\phi$  and  $\chi$ . Input and output are strided arrays of dense complex data with spacing between adjacent complex values given as *cstri* and *cstro* elements respectively. Thus, *Co* for example is packed as:

$$[\Re(C_{00}) \ \Im(C_{00}) \dots (2 \times cstro) \dots \Re(C_{0,-1}) \ \Im(C_{0,-1})]$$

This stride system allows for packing data more conveniently for upward and downward passes in the FMM proper.

#### Parameters

<i>Co</i>	on output contains rotated coefficients;
<i>cstro</i>	stride in <i>Co</i> , in number of complex elements;
<i>Ci</i>	input coefficients, to be rotated;
<i>cstri</i>	stride in <i>Ci</i> , in number of complex elements;
<i>N</i>	maximum order of coefficients;
<i>nq</i>	number of source terms;
<i>H</i>	rotation coefficients for angle $\theta$ , from <b>wbfmm_coefficients_H_rotation</b> (p. 37)(...);
<i>ph</i>	angle $\phi$ for rotation;
<i>ch</i>	angle $\chi$ for rotation.

#### Returns

0 on success

### 5.5.2.4 `gint wbfmm_laplace_rotate_H_f ( gfloat * Co, gint cstro, gfloat * Ci, gint cstri, gint N, gint nq, gfloat * H, gfloat ph, gfloat ch )`

Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.

Given the rotation coefficients  $H$  for angle  $\theta$  from **wbfmm\_coefficients\_H\_rotation\_f** (p. 38)(...), rotate input coefficients to new system of axes, using  $H$  and angles  $\phi$  and  $\chi$ . Input and output are strided arrays of dense complex data with spacing between adjacent complex values given as *cstri* and *cstro* elements respectively. Thus, *Co* for example is packed as:

$$[\Re(C_{00}) \ \Im(C_{00}) \dots (2 \times cstro) \dots \Re(C_{0,-1}) \ \Im(C_{0,-1})]$$

This stride system allows for packing data more conveniently for upward and downward passes in the FMM proper.



## Parameters

<i>Co</i>	on output contains rotated coefficients;
<i>cstro</i>	stride in <i>Co</i> , in number of complex elements;
<i>Ci</i>	input coefficients, to be rotated;
<i>cstri</i>	stride in <i>Ci</i> , in number of complex elements;
<i>N</i>	maximum order of coefficients;
<i>nq</i>	number of source terms;
<i>H</i>	rotation coefficients for angle $\theta$ , from <b>wbfmm_coefficients_H_rotation_f</b> (p. 38)(...);
<i>ph</i>	angle $\phi$ for rotation;
<i>ch</i>	angle $\chi$ for rotation.

## Returns

0 on success

**5.5.2.5** `gint wbfmm_rotate_H ( gdouble * Co, gint cstro, gdouble * Ci, gint cstri, gint N, gdouble * H, gdouble ph, gdouble ch )`

Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients.

Given the rotation coefficients *H* for angle  $\theta$  from **wbfmm\_coefficients\_H\_rotation** (p. 37)(...), rotate input coefficients to new system of axes, using *H* and angles  $\phi$  and  $\chi$ . Input and output are strided arrays of dense complex data with spacing between adjacent complex values given as *cstri* and *cstro* elements respectively. Thus, *Co* for example is packed as:

$$[\Re(C_{00}) \quad \Im(C_{00}) \dots (2 \times cstro) \dots \Re(C_{0,-1}) \quad \Im(C_{0,-1})]$$

This stride system allows for packing data more conveniently for upward and downward passes in the FMM proper.

The function is available as a reference version `wbfmm_rotate_H_ref(...)` and an optimized version `wbfmm_rotate_H_avx(...)` which uses AVX optimizations if available. The compile time switch `-DWBFMM_USE_AVX` selects the AVX version.

## Parameters

<i>Co</i>	on output contains rotated coefficients;
<i>cstro</i>	stride in <i>Co</i> , in number of complex elements;
<i>Ci</i>	input coefficients, to be rotated;
<i>cstri</i>	stride in <i>Ci</i> , in number of complex elements;
<i>N</i>	maximum order of coefficients;
<i>H</i>	rotation coefficients for angle $\theta$ , from <b>wbfmm_coefficients_H_rotation</b> (p. 37)(...);
<i>ph</i>	angle $\phi$ for rotation;
<i>ch</i>	angle $\chi$ for rotation.

## Returns

0 on success

**5.5.2.6** `gint wbfmm_rotate_H_f ( gfloat * Co, gint cstro, gfloat * Ci, gint cstri, gint N, gfloat * H, gfloat ph, gfloat ch )`

Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients.

Given the rotation coefficients *H* for angle  $\theta$  from **wbfmm\_coefficients\_H\_rotation\_f** (p. 38)(...), rotate input coefficients to new system of axes, using *H* and angles  $\phi$  and  $\chi$ . Input and output are strided arrays of dense complex data with spacing between adjacent complex values given as *cstri* and *cstro* elements respectively. Thus, *Co* for example is packed as:

$$[\Re(C_{00}) \quad \Im(C_{00}) \dots (2 \times cstro) \dots \Re(C_{0,-1}) \quad \Im(C_{0,-1})]$$

This stride system allows for packing data more conveniently for upward and downward passes in the FMM proper.

The function is available as a reference version `wbfmm_rotate_H_ref_f(...)` and an optimized version `wbfmm_rotate_H_avx_f(...)` which uses AVX optimizations if available. The compile time switch `-DWBMM_USE_AVX` selects the AVX version.

#### Parameters

<i>Co</i>	on output contains rotated coefficients;
<i>cstro</i>	stride in <i>Co</i> , in number of complex elements;
<i>Ci</i>	input coefficients, to be rotated;
<i>cstri</i>	stride in <i>Ci</i> , in number of complex elements;
<i>N</i>	maximum order of coefficients;
<i>H</i>	rotation coefficients for angle $\theta$ , from <code>wbfmm_coefficients_H_rotation_f</code> (p. 38)(...);
<i>ph</i>	angle $\phi$ for rotation;
<i>ch</i>	angle $\chi$ for rotation.

#### Returns

0 on success

**5.5.2.7** `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  $(\theta, \phi, \chi)$  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

<i>ix</i>	initial coordinate system $x$ axis;
<i>iy</i>	initial coordinate system $y$ axis;
<i>iz</i>	initial coordinate system $z$ axis;
<i>jx</i>	rotated coordinate system $x$ axis;
<i>jy</i>	rotated coordinate system $y$ axis;
<i>jz</i>	rotated coordinate system $z$ axis;
<i>th</i>	on exit, $\theta$ for rotation;
<i>ph</i>	on exit, $\phi$ for rotation;
<i>ch</i>	on exit, $\chi$ for rotation.

#### Returns

0 on success

**5.5.2.8** `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  $(\theta, \phi, \chi)$  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

<i>ix</i>	initial coordinate system $x$ axis;
<i>iy</i>	initial coordinate system $y$ axis;
<i>iz</i>	initial coordinate system $z$ axis;
<i>jx</i>	rotated coordinate system $x$ axis;
<i>jy</i>	rotated coordinate system $y$ axis;
<i>jz</i>	rotated coordinate system $z$ axis;
<i>th</i>	on exit, $\theta$ for rotation;
<i>ph</i>	on exit, $\phi$ for rotation;
<i>ch</i>	on exit, $\chi$ for rotation.

## Returns

0 on success

## 5.6 Translation operators

Translation of expansions.

### Functions

- gint **wbfmm\_coefficients\_SR\_coaxial** (gdouble \*cfftSR, gint L, gdouble kr, gdouble \*work)  
*Generate coefficients for coaxial singular-to-regular translation.*
- gint **wbfmm\_coaxial\_translate** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gdouble \*cfft, gint L, gboolean complex)  
*Perform coaxial translation of multipole expansion.*
- gint **wbfmm\_coefficients\_SR\_coaxial\_f** (gfloat \*cfftSR, gint L, gfloat kr, gfloat \*work)  
*Generate coefficients for coaxial singular-to-regular translation.*
- gint **wbfmm\_coaxial\_translate\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gfloat \*cfft, gint L, gboolean complex)  
*Perform coaxial translation of multipole expansion.*
- gint **wbfmm\_coefficients\_RR\_coaxial** (gdouble \*cfftRR, gint L, gdouble kr, gdouble \*work)  
*Generate coefficients for coaxial regular-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.*

### 5.6.1 Detailed Description

Translation of expansions.

### 5.6.2 Function Documentation

**5.6.2.1** gint wbfmm\_coaxial\_translate ( gdouble \* Co, gint cstro, gint No, gdouble \* Ci, gint cstri, gint Ni, gdouble \* cfft, gint L, gboolean complex )

Perform coaxial translation of multipole expansion.

Compute the coaxial translation of a multipole expansion along its  $z$  axis, using coefficients from **wbfmm\_coefficients\_SR\_coaxial** (p.44)(...) (complex) or **wbfmm\_coefficients\_RR\_coaxial** (p.43)(...) (real). Input and output coefficients are strided data as described for **wbfmm\_rotate\_H** (p.39)(...).

#### Parameters

<i>Co</i>	on output contains translated multipole expansion;
<i>cstro</i>	stride for output data in number of complex elements;
<i>No</i>	order of output expansion;
<i>Ci</i>	input multipole expansion;
<i>cstri</i>	stride for input data in number of complex elements;
<i>Ni</i>	order of input expansion;
<i>cfft</i>	translation coefficients;
<i>L</i>	maximum order of translation coefficients;
<i>complex</i>	if TRUE treat <i>cfft</i> as complex (e.g. for singular-to-regular translation); if FALSE treat as real (e.g. regular-to-regular or singular-to-singular).

#### Returns

0 on success

**5.6.2.2** `gint wbfmm_coaxial_translate_f ( gfloat * Co, gint cstro, gint No, gfloat * Ci, gint cstri, gint Ni, gfloat * cfft, gint L, gboolean complex )`

Perform coaxial translation of multipole expansion.

Compute the coaxial translation of a multipole expansion along its  $z$  axis, using coefficients from **wbfmm\_**[coefficients\\_SR\\_coaxial\\_f](#) (p. 44)(...) (complex) or **wbfmm\_**[coefficients\\_RR\\_coaxial\\_f](#) (p. 43)(...) (real). Input and output coefficients are strided data as described for **wbfmm\_**[rotate\\_H\\_f](#) (p. 39)(...).

Parameters

<i>Co</i>	on output contains translated multipole expansion;
<i>cstro</i>	stride for output data in number of complex elements;
<i>No</i>	order of output expansion;
<i>Ci</i>	input multipole expansion;
<i>cstri</i>	stride for input data in number of complex elements;
<i>Ni</i>	order of input expansion;
<i>cfft</i>	translation coefficients;
<i>L</i>	maximum order of translation coefficients;
<i>complex</i>	if TRUE treat <i>cfft</i> as complex (e.g. for singular-to-regular translation); if FALSE treat as real (e.g. regular-to-regular or singular-to-singular).

Returns

0 on success

**5.6.2.3** `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

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

Returns

0 on success

**5.6.2.4** `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

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

**Returns**

0 on success

#### 5.6.2.5 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**

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

**Returns**

0 on success

#### 5.6.2.6 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**

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

**Returns**

0 on success

## 5.7 Utility and convenience functions

Various functions of use in debugging or underlying utilities.

### Functions

- gint **wbfmm\_cartesian\_to\_spherical** (gdouble \*x0, gdouble \*x, gdouble \*r, gdouble \*th, gdouble \*ph)  
*Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .*
- 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\_total\_field** (gdouble k, gdouble \*xs, gint xstride, gdouble \*src, gint sstride, gint nsrc, gdouble \*xf, gdouble \*field)  
*Compute total field 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 idx, guint32 level, gdouble \*x0, gdouble D, gdouble \*x, gdouble \*wb)  
*Find the coordinates of a box from its Morton index.*
- gint **wbfmm\_tree\_box\_centre** (wbfmm\_tree\_t \*t, guint32 level, guint64 b, gdouble \*xb, gdouble \*wb)  
*Find centre and width of box in an octree.*
- gint **wbfmm\_points\_origin\_width** (gdouble \*x, gint str, gint n, gdouble \*xmin, gdouble \*D, gboolean init\_↔ limits)  
*Find limits of a cube containing a set of points.*
- 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\_cartesian\_to\_spherical\_f** (gfloat \*x0, gfloat \*x, gfloat \*r, gfloat \*th, gfloat \*ph)  
*Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .*
- 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\_total\_field\_f** (gfloat k, gfloat \*xs, gint xstride, gfloat \*src, gint sstride, gint nsrc, gfloat \*xf, gfloat \*field)

*Compute total field 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 idx, guint32 level, gfloat \*x0, gfloat D, gfloat \*x, gfloat \*wb)

*Find the coordinates of a box from its Morton index.*

- gint **wbfmm\_tree\_box\_centre\_f** (wbfmm\_tree\_t \*t, guint32 level, guint64 b, gfloat \*xb, gfloat \*wb)

*Find centre and width of box in an octree.*

- gint **wbfmm\_points\_origin\_width\_f** (gfloat \*x, gint str, gint n, gfloat \*xmin, gfloat \*D, gboolean init\_limits)

*Find limits of a cube containing a set of points.*

- 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**

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

**Returns**

0 on success

**5.7.2.9** `gint wbfmm_box_location_from_index ( quint64 idx, quint32 level, gdouble * x0, gdouble D, gdouble * x, gdouble * wb )`

Find the coordinates of a box from its Morton index.

**Parameters**

<i>idx</i>	Morton index of box;
<i>level</i>	level in octree of box;
<i>x0</i>	origin of top-level box;
<i>D</i>	width of top-level box;
<i>x</i>	coordinates of box <i>idx</i> at level <i>level</i> ;
<i>wb</i>	width of box at level <i>level</i>

**Returns**

0 on success

**5.7.2.10** `gint wbfmm_box_location_from_index_f ( quint64 idx, quint32 level, gfloat * x0, gfloat D, gfloat * x, gfloat * wb )`

Find the coordinates of a box from its Morton index.

**Parameters**

<i>idx</i>	Morton index of box;
<i>level</i>	level in octree of box;
<i>x0</i>	origin of top-level box;
<i>D</i>	width of top-level box;
<i>x</i>	coordinates of box <i>idx</i> at level <i>level</i> ;
<i>wb</i>	width of box at level <i>level</i>

**Returns**

0 on success

**5.7.2.11** `gint wbfmm_cartesian_to_spherical ( gdouble * x0, gdouble * x, gdouble * r, gdouble * th, gdouble * ph )`

Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .

**Parameters**

<i>x0</i>	centre of coordinate system;
-----------	------------------------------

$x$	point whose coordinates are to be found;
$r$	$r$ ;
$th$	$\theta$ ;
$ph$	$\phi$

**Returns**

0 on success

#### 5.7.2.12 `gint wbfmm_cartesian_to_spherical_f ( gfloat * x0, gfloat * x, gfloat * r, gfloat * th, gfloat * ph )`

Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .

**Parameters**

$x0$	centre of coordinate system;
$x$	point whose coordinates are to be found;
$r$	$r$ ;
$th$	$\theta$ ;
$ph$	$\phi$

**Returns**

0 on success

#### 5.7.2.13 `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

#### 5.7.2.14 `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;

<i>y</i>	point coordinates in new axes
----------	-------------------------------

**Returns**

0 on success

**5.7.2.15 gint wbfmm\_legendre\_init ( gdouble C, gdouble S, gdouble \* P0, gdouble \* P10, gdouble \* P11 )**

Initialize normalized associated Legendre functions.

**Parameters**

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

**Returns**

0 on success

**5.7.2.16 gint wbfmm\_legendre\_init\_f ( gfloat C, gfloat S, gfloat \* P0, gfloat \* P10, gfloat \* P11 )**

Initialize normalized associated Legendre functions.

**Parameters**

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

**Returns**

0 on success

**5.7.2.17 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**

<i>Pnm1</i>	pointer to array of normalized associated Legendre functions for $n-1$ ;
<i>Pn</i>	pointer to array of normalized associated Legendre functions for $n$ ;
<i>n</i>	order of $Pn$ ;

<i>C</i>	$\cos \theta$ ;
<i>S</i>	$\sin \theta$ ;

**Returns**

0 on success

**5.7.2.18 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**

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

**Returns**

0 on success

**5.7.2.19 gint wbfmm\_points\_origin\_width ( gdouble \* *x*, gint *str*, gint *n*, gdouble \* *xmin*, gdouble \* *D*, gboolean *init\_limits* )**

Find limits of a cube containing a set of points.

**Parameters**

<i>x</i>	array of points coordinates;
<i>str</i>	stride of points in <i>x</i> ;
<i>n</i>	number of points in <i>x</i> ;
<i>xmin</i>	origin of cube containing all points in <i>x</i> ;
<i>D</i>	width of cube containing all points in <i>x</i> ;
<i>init_limits</i>	if TRUE initialize limits overwriting any data in <i>xmin</i>

**Returns**

0 on success

**5.7.2.20 gint wbfmm\_points\_origin\_width\_f ( gfloat \* *x*, gint *str*, gint *n*, gfloat \* *xmin*, gfloat \* *D*, gboolean *init\_limits* )**

Find limits of a cube containing a set of points.

**Parameters**

<i>x</i>	array of points coordinates;
----------	------------------------------

<i>str</i>	stride of points in <i>x</i> ;
<i>n</i>	number of points in <i>x</i> ;
<i>xmin</i>	origin of cube containing all points in <i>x</i> ;
<i>D</i>	width of cube containing all points in <i>x</i> ;
<i>init_limits</i>	if TRUE initialize limits overwriting any data in <i>xmin</i>

**Returns**

0 on success

**5.7.2.21 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. 53)(...) and **wbfmm\_rotation\_angles** (p. 40)(...)

**Parameters**

<i>xi</i>	origin of shift;
<i>xj</i>	destination of shift;
<i>th</i>	$\theta$ for shift;
<i>ph</i>	$\phi$ for shift;
<i>ch</i>	$\chi$ for shift;
<i>r</i>	distance between source and destination points

**Returns**

0 on success

**5.7.2.22 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. 54)(...) and **wbfmm\_rotation\_angles\_f** (p. 40)(...)

**Parameters**

<i>xi</i>	origin of shift;
<i>xj</i>	destination of shift;
<i>th</i>	$\theta$ for shift;
<i>ph</i>	$\phi$ for shift;
<i>ch</i>	$\chi$ for shift;
<i>r</i>	distance between source and destination points

**Returns**

0 on success

**5.7.2.23 gint wbfmm\_shift\_coordinates ( gdouble \* *x*, gdouble \* *y*, gdouble \* *ix*, gdouble \* *iy*, gdouble \* *iz*, gdouble \* *r* )**

Find system of axes for coordinate shift.

## Parameters

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

## Returns

0 on success

**5.7.2.24** `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

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

## Returns

0 on success

**5.7.2.25** `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  $\mathbf{x}_n \sum_{n=1}^N \mathbf{f}_n \cdot \nabla h_0(\mathbf{x} - \mathbf{x}_n) / 4\pi$ .

## Parameters

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

## Returns

0 on success

**5.7.2.26** `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  $\mathbf{x}_n \sum_{n=1}^N \mathbf{f}_n \cdot \nabla h_0(\mathbf{x} - \mathbf{x}_n) / 4\pi$ .



## Parameters

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

## Returns

0 on success

5.7.2.27 `gint wbfmm_total_field ( gdouble k, gdouble * xs, gint xstride, gdouble * src, gint sstride, gint nsrc, gdouble * xf, gdouble * field )`

Compute total field by direct evaluation.

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

## Parameters

<i>k</i>	wavenumber;
<i>xs</i>	array of source positions;
<i>xstride</i>	stride in <i>xs</i> between source positions;
<i>src</i>	array of complex scalar source strengths;
<i>sstride</i>	stride in <i>src</i> ;
<i>nsrc</i>	number of sources;
<i>xf</i>	point for field evaluation;
<i>field</i>	incremented with computed field

## Returns

0 on success

5.7.2.28 `gint wbfmm_total_field_f ( gfloat k, gfloat * xs, gint xstride, gfloat * src, gint sstride, gint nsrc, gfloat * xf, gfloat * field )`

Compute total field by direct evaluation.

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

## Parameters

<i>k</i>	wavenumber;
<i>xs</i>	array of source positions;
<i>xstride</i>	stride in <i>xs</i> between source positions;
<i>src</i>	array of complex scalar source strengths;
<i>sstride</i>	stride in <i>src</i> ;
<i>nsrc</i>	number of sources;
<i>xf</i>	point for field evaluation;

<i>field</i>	incremented with computed field
--------------	---------------------------------

**Returns**

0 on success

**5.7.2.29** `gint wbfmm_tree_box_centre ( wbfmm_tree_t * t, guint32 level, guint64 b, gdouble * xb, gdouble * wb )`

Find centre and width of box in an octree.

**Parameters**

<i>t</i>	an octree;
<i>level</i>	level inside <i>t</i> ;
<i>b</i>	Morton index of box at level <i>level</i> ;
<i>xb</i>	centre of box with index <i>b</i> at level <i>level</i> ;
<i>wb</i>	width of box at level <i>level</i> ;

**Returns**

0 on success

**5.7.2.30** `gint wbfmm_tree_box_centre_f ( wbfmm_tree_t * t, guint32 level, guint64 b, gfloat * xb, gfloat * wb )`

Find centre and width of box in an octree.

**Parameters**

<i>t</i>	an octree;
<i>level</i>	level inside <i>t</i> ;
<i>b</i>	Morton index of box at level <i>level</i> ;
<i>xb</i>	centre of box with index <i>b</i> at level <i>level</i> ;
<i>wb</i>	width of box at level <i>level</i> ;

**Returns**

0 on success

**5.7.2.31** `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**

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

**Returns**

0 on success

5.7.2.32 `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**

<i>t</i>	an octree with a list of sources attached;
<i>q</i>	source strengths (if NULL, source strengths are not written);
<i>stride</i>	source strength stride in <i>q</i> ;
<i>f</i>	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\_laplace\_expansion\_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\_laplace\_coaxial\_translate\_SS** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)  
*Singular to singular translation for Laplace expansion.*
- gint **wbfmm\_laplace\_child\_parent\_shift** (gdouble \*Cp, gint Np, gdouble \*Cc, gint Nc, gint nq, gdouble \*H03, gdouble \*H47, gint Lh, gdouble wb, gdouble \*work)  
*Upward shift of singular expansion from eight children to common parent in Laplace problem.*
- gint **wbfmm\_laplace\_parent\_child\_shift** (gdouble \*Cc, gint Nc, gdouble \*Cp, gint Np, gint nq, gdouble \*H03, gdouble \*H47, gint Lh, gdouble wb, gdouble \*work)  
*Downward shift of regular expansion from parent to eight children in Laplace problem.*
- gint **wbfmm\_laplace\_field\_coefficients** (gdouble \*x, gint N, gboolean grad, gdouble \*cfft, gdouble \*work)  
*Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)*
- gint **wbfmm\_laplace\_expansion\_apply** (gdouble \*C, gint cstr, gint nq, gdouble \*ec, gint N, gdouble \*f)  
*Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using **wbfmm\_laplace\_field\_coefficients** or **wbfmm\_laplace\_local\_coefficients** for the field (singular) or local (regular) expansions respectively.*
- gint **wbfmm\_laplace\_local\_coefficients** (gdouble \*x, gint N, gboolean grad, gdouble \*cfft, gdouble \*work)  
*Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)*
- gint **wbfmm\_box\_fields\_laplace** (**wbfmm\_tree\_t** \*t, gint level, gdouble \*xf, gdouble \*field, gdouble \*work)  
*Evaluate the Laplace field generated by all boxes on a given level of an octree.*
- gint **wbfmm\_laplace\_coaxial\_translate\_init** (gint N)  
*Initialize lookup tables of Laplace translation coefficients.*
- gint **wbfmm\_laplace\_coaxial\_translate\_SR** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)  
*Singular to regular translation for Laplace expansion.*
- gint **wbfmm\_laplace\_coaxial\_translate\_RR** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)  
*Regular to regular translation for Laplace expansion.*
- gint **wbfmm\_laplace\_field** (gdouble \*xs, gint xstride, gdouble \*src, gint sstride, gint nq, gdouble \*normals, gint nstr, gdouble \*dipoles, gint dstr, gint nsrc, gdouble \*xf, gdouble \*field)
- gint **wbfmm\_laplace\_expansion\_local\_evaluate** (gdouble \*x0, gdouble \*cfft, gint cstr, gint N, gint nq, gdouble \*xf, gdouble \*field, gdouble \*work)
- gint **wbfmm\_laplace\_expansion\_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\_laplace\_coaxial\_translate\_SS\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Singular to singular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_child\_parent\_shift\_f** (gfloat \*Cp, gint Np, gfloat \*Cc, gint Nc, gint nq, gfloat \*H03, gfloat \*H47, gint Lh, gfloat wb, gfloat \*work)

*Upward shift of singular expansion from eight children to common parent in Laplace problem.*

- gint **wbfmm\_laplace\_parent\_child\_shift\_f** (gfloat \*Cc, gint Nc, gfloat \*Cp, gint Np, gint nq, gfloat \*H03, gfloat \*H47, gint Lh, gfloat wb, gfloat \*work)

*Downward shift of regular expansion from parent to eight children in Laplace problem.*

- gint **wbfmm\_laplace\_field\_coefficients\_f** (gfloat \*x, gint N, gboolean grad, gfloat \*cfft, gfloat \*work)

*Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)*

- gint **wbfmm\_laplace\_expansion\_apply\_f** (gfloat \*C, gint cstr, gint nq, gfloat \*ec, gint N, gfloat \*f)

*Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using **wbfmm\_laplace\_field\_coefficients** or **wbfmm\_laplace\_local\_coefficients** for the field (singular) or local (regular) expansions respectively.*

- gint **wbfmm\_laplace\_local\_coefficients\_f** (gfloat \*x, gint N, gboolean grad, gfloat \*cfft, gfloat \*work)

*Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)*

- gint **wbfmm\_box\_fields\_laplace\_f** (**wbfmm\_tree\_t** \*t, gint level, gfloat \*xf, gfloat \*field, gfloat \*work)

*Evaluate the Laplace field generated by all boxes on a given level of an octree.*

- gint **wbfmm\_laplace\_coaxial\_translate\_init\_f** (gint N)

*Initialize lookup tables of Laplace translation coefficients.*

- gint **wbfmm\_laplace\_coaxial\_translate\_SR\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Singular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_RR\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Regular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_field\_f** (gfloat \*xs, gint xstride, gfloat \*src, gint sstride, gint nq, gfloat \*normals, gint nstr, gfloat \*dipoles, gint dstr, gint nsrc, gfloat \*xf, gfloat \*field)
- gint **wbfmm\_laplace\_expansion\_local\_evaluate\_f** (gfloat \*x0, gfloat \*cfft, gint cstr, gint N, gint nq, gfloat \*xf, gfloat \*field, gfloat \*work)

### 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\_box\_fields\_laplace ( wbfmm\_tree\_t \* t, gint level, gdouble \* xf, gdouble \* field, gdouble \* work )

Evaluate the Laplace field generated by all boxes on a given level of an octree.

Parameters

<i>t</i>	octree;
<i>level</i>	level at which to use expansions;
<i>xf</i>	field evaluation point;
<i>field</i>	on output contains the sum of singular expansions from each box on level <i>level</i> evaluated at <i>xf</i> ;

<i>work</i>	workspace.
-------------	------------

**Returns**

0 on success

### 5.8.2.2 `gint wbfmm_box_fields_laplace_f ( wbfmm_tree_t * t, gint level, gfloat * xf, gfloat * field, gfloat * work )`

Evaluate the Laplace field generated by all boxes on a given level of an octree.

**Parameters**

<i>t</i>	octree;
<i>level</i>	level at which to use expansions;
<i>xf</i>	field evaluation point;
<i>field</i>	on output contains the sum of singular expansions from each box on level <i>level</i> evaluated at <i>xf</i> ;
<i>work</i>	workspace.

**Returns**

0 on success

### 5.8.2.3 `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**

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

**Returns**

0 on success

### 5.8.2.4 `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**

<i>x0</i>	origin of expansion;
-----------	----------------------

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

**Returns**

0 on success

**5.8.2.5** `gint wbfmm_laplace_child_parent_shift ( gdouble * Cp, gint Np, gdouble * Cc, gint Nc, gint nq, gdouble * H03, gdouble * H47, gint Lh, gdouble wb, gdouble * work )`

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

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**

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;
<i>nq</i>	number of elements in source;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>H47</i>	rotation coefficients for 'upper' children (Morton index 4-7);
<i>Lh</i>	maximum order of rotation coefficients;
<i>wb</i>	child box width;
<i>work</i>	workspace

**Returns**

0 on success

**5.8.2.6** `gint wbfmm_laplace_child_parent_shift_f ( gfloat * Cp, gint Np, gfloat * Cc, gint Nc, gint nq, gfloat * H03, gfloat * H47, gint Lh, gfloat wb, gfloat * work )`

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

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**

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;

<i>nq</i>	number of elements in source;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>H47</i>	rotation coefficients for 'upper' children (Morton index 4-7);
<i>Lh</i>	maximum order of rotation coefficients;
<i>wb</i>	child box width;
<i>work</i>	workspace

**Returns**

0 on success

**5.8.2.7 gint wbfmm\_laplace\_coaxial\_translate\_init ( gint *N* )**

Initialize lookup tables of Laplace translation coefficients.

Initialize lookup tables of Laplace translation coefficients for use in coaxial translation of Laplace expansions. This function must be called before any coaxial translation is performed in a Laplace problem.

**Parameters**

<i>N</i>	maximum order of expansion to be translated.
----------	--

**Returns**

0 on success

**5.8.2.8 gint wbfmm\_laplace\_coaxial\_translate\_init\_f ( gint *N* )**

Initialize lookup tables of Laplace translation coefficients.

Initialize lookup tables of Laplace translation coefficients for use in coaxial translation of Laplace expansions. This function must be called before any coaxial translation is performed in a Laplace problem.

**Parameters**

<i>N</i>	maximum order of expansion to be translated.
----------	--

**Returns**

0 on success

**5.8.2.9 gint wbfmm\_laplace\_coaxial\_translate\_RR ( gdouble \* *Co*, gint *cstro*, gint *No*, gdouble \* *Ci*, gint *cstri*, gint *Ni*, gint *nq*, gdouble *t* )**

Regular to regular translation for Laplace expansion.

Translate a regular expansion for the Laplace problem along the  $z$  axis to a regular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init** (p.62)(...)

**Parameters**

<i>Co</i>	on output, expansion about new centre;
-----------	--



<i>cstro</i>	stride in <i>Co</i> ;
<i>No</i>	order of expansion in <i>Co</i> ;
<i>Ci</i>	input expansion coefficients;
<i>cstri</i>	stride in <i>Ci</i> ;
<i>Ni</i>	order of expansion in <i>Ci</i> ;
<i>nq</i>	number of source terms in expansion;
<i>t</i>	distance to translate expansion.

**Returns**

0 on success

**5.8.2.10** `gint wbfmm_laplace_coaxial_translate_RR_f ( gfloat * Co, gint cstro, gint No, gfloat * Ci, gint cstri, gint Ni, gint nq, gfloat t )`

Regular to regular translation for Laplace expansion.

Translate a regular expansion for the Laplace problem along the  $z$  axis to a regular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init\_f** (p. 62)(...)

**Parameters**

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

**Returns**

0 on success

**5.8.2.11** `gint wbfmm_laplace_coaxial_translate_SR ( gdouble * Co, gint cstro, gint No, gdouble * Ci, gint cstri, gint Ni, gint nq, gdouble t )`

Singular to regular translation for Laplace expansion.

Translate a singular expansion for the Laplace problem along the  $z$  axis to a regular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init** (p. 62)(...)

**Parameters**

<i>Co</i>	on output, expansion about new centre;
<i>cstro</i>	stride in <i>Co</i> ;
<i>No</i>	order of expansion in <i>Co</i> ;
<i>Ci</i>	input expansion coefficients;
<i>cstri</i>	stride in <i>Ci</i> ;

$Ni$	order of expansion in $Ci$ ;
$nq$	number of source terms in expansion;
$t$	distance to translate expansion.

**Returns**

0 on success

**5.8.2.12** `gint wbfmm_laplace_coaxial_translate_SR_f ( gfloat *  $Co$ , gint  $cstro$ , gint  $No$ , gfloat *  $Ci$ , gint  $cstri$ , gint  $Ni$ , gint  $nq$ , gfloat  $t$  )`

Singular to regular translation for Laplace expansion.

Translate a singular expansion for the Laplace problem along the  $z$  axis to a regular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init\_f** (p. 62)(...)

**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

**5.8.2.13** `gint wbfmm_laplace_coaxial_translate_SS ( gdouble *  $Co$ , gint  $cstro$ , gint  $No$ , gdouble *  $Ci$ , gint  $cstri$ , gint  $Ni$ , gint  $nq$ , gdouble  $t$  )`

Singular to singular translation for Laplace expansion.

Translate a singular expansion for the Laplace problem along the  $z$  axis to a singular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init** (p. 62)(...)

**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

5.8.2.14 `gint wbfmm_laplace_coaxial_translate_SS_f ( gfloat * Co, gint cstro, gint No, gfloat * Ci, gint cstri, gint Ni, gint nq, gfloat t )`

Singular to singular translation for Laplace expansion.

Translate a singular expansion for the Laplace problem along the  $z$  axis to a singular expansion about a new centre. Before any Laplace translation function is called, the translation coefficients must be initialized with a call to **wbfmm\_laplace\_coaxial\_translate\_init\_f** (p. 62)(...)

Parameters

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

Returns

0 on success

5.8.2.15 `gint wbfmm_laplace_expansion_apply ( gdouble * C, gint cstr, gint nq, gdouble * ec, gint N, gdouble * f )`

Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using `wbfmm_laplace_field_coefficients`) or `wbfmm_laplace_local_coefficients`) for the field (singular) or local (regular) expansions respectively.

Parameters

<i>C</i>	coefficients of expansion;
<i>cstr</i>	stride in <i>C</i> ;
<i>nq</i>	number of source terms in <i>C</i> ;
<i>ec</i>	evaluation coefficients;
<i>N</i>	order of expansion;
<i>f</i>	on exit contains evaluated field.

Returns

0 on success

5.8.2.16 `gint wbfmm_laplace_expansion_apply_f ( gfloat * C, gint cstr, gint nq, gfloat * ec, gint N, gfloat * f )`

Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using `wbfmm_laplace_field_coefficients`) or `wbfmm_laplace_local_coefficients`) for the field (singular) or local (regular) expansions respectively.

Parameters

<i>C</i>	coefficients of expansion;
<i>cstr</i>	stride in <i>C</i> ;

<i>nq</i>	number of source terms in <i>C</i> ;
<i>ec</i>	evaluation coefficients;
<i>N</i>	order of expansion;
<i>f</i>	on exit contains evaluated field.

#### Returns

0 on success

**5.8.2.17** `gint wbfmm_laplace_expansion_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

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

#### Returns

0 on success

**5.8.2.18** `gint wbfmm_laplace_expansion_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

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

#### Returns

0 on success

**5.8.2.19** `gint wbfmm_laplace_expansion_local_evaluate ( gdouble * x0, gdouble * cfft, gint cstr, gint N, gint nq, gdouble * xf, gdouble * field, gdouble * work )`

## Parameters

--	--

5.8.2.20 `gint wbfmm_laplace_expansion_local_evaluate_f ( gfloat * x0, gfloat * cfft, gint cstr, gint N, gint nq, gfloat * xf, gfloat * field, gfloat * work )`

## Parameters

--	--

5.8.2.21 `gint wbfmm_laplace_field ( gdouble * xs, gint xstride, gdouble * src, gint sstride, gint nq, gdouble * normals, gint nstr, gdouble * dipoles, gint dstr, gint nsr, gdouble * xf, gdouble * field )`

## Parameters

--	--

5.8.2.22 `gint wbfmm_laplace_field_coefficients ( gdouble * x, gint N, gboolean grad, gdouble * cfft, gdouble * work )`

Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)

## Parameters

<i>x</i>	location of evaluation point relative to centre of expansion;
<i>N</i>	order of expansion;
<i>grad</i>	if TRUE generate coefficients for gradient of field;
<i>cfft</i>	on exit contains evaluation coefficients;
<i>work</i>	workspace.

## Returns

0 on success

5.8.2.23 `gint wbfmm_laplace_field_coefficients_f ( gfloat * x, gint N, gboolean grad, gfloat * cfft, gfloat * work )`

Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)

## Parameters

<i>x</i>	location of evaluation point relative to centre of expansion;
<i>N</i>	order of expansion;
<i>grad</i>	if TRUE generate coefficients for gradient of field;
<i>cfft</i>	on exit contains evaluation coefficients;
<i>work</i>	workspace.

## Returns

0 on success

5.8.2.24 `gint wbfmm_laplace_field_f ( gfloat * xs, gint xstride, gfloat * src, gint sstride, gint nq, gfloat * normals, gint nstr, gfloat * dipoles, gint dstr, gint nsr, gfloat * xf, gfloat * field )`

## Parameters

--	--

#### 5.8.2.25 `gint wbfmm_laplace_local_coefficients ( gdouble * x, gint N, gboolean grad, gdouble * cfft, gdouble * work )`

Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)

## Parameters

<i>x</i>	location of evaluation point relative to centre of expansion;
<i>N</i>	order of expansion;
<i>grad</i>	if TRUE generate coefficients for gradient of field;
<i>cfft</i>	on exit contains evaluation coefficients;
<i>work</i>	workspace.

## Returns

0 on success

#### 5.8.2.26 `gint wbfmm_laplace_local_coefficients_f ( gfloat * x, gint N, gboolean grad, gfloat * cfft, gfloat * work )`

Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)

## Parameters

<i>x</i>	location of evaluation point relative to centre of expansion;
<i>N</i>	order of expansion;
<i>grad</i>	if TRUE generate coefficients for gradient of field;
<i>cfft</i>	on exit contains evaluation coefficients;
<i>work</i>	workspace.

## Returns

0 on success

#### 5.8.2.27 `gint wbfmm_laplace_parent_child_shift ( gdouble * Cc, gint Nc, gdouble * Cp, gint Np, gint nq, gdouble * H03, gdouble * H47, gint Lh, gdouble wb, gdouble * work )`

Downward shift of regular expansion from parent to eight children in Laplace problem.

Shift the expansion of a parent box to its eight child boxes. 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. Note that rotation coefficients *H03* and *H47* are the same as for the upward pass but switched (because the rotations are performed in the opposite direction).

## Parameters

<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;

<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>nq</i>	number of elements in source;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>H47</i>	rotation coefficients for 'upper' children (Morton index 4-7);
<i>Lh</i>	maximum order of rotation coefficients;
<i>wb</i>	parent box width;
<i>work</i>	workspace

**Returns**

0 on success

**5.8.2.28** `gint wbfmm_laplace_parent_child_shift_f ( gfloat * Cc, gint Nc, gfloat * Cp, gint Np, gint nq, gfloat * H03, gfloat * H47, gint Lh, gfloat wb, gfloat * work )`

Downward shift of regular expansion from parent to eight children in Laplace problem.

Shift the expansion of a parent box to its eight child boxes. 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. Note that rotation coefficients *H03* and *H47* are the same as for the upward pass but switched (because the rotations are performed in the opposite direction).

**Parameters**

<i>Cc</i>	child expansion array;
<i>Nc</i>	order of child expansions;
<i>Cp</i>	parent expansion array;
<i>Np</i>	order of parent expansion;
<i>nq</i>	number of elements in source;
<i>H03</i>	rotation coefficients for 'lower' children (Morton index 0-3);
<i>H47</i>	rotation coefficients for 'upper' children (Morton index 4-7);
<i>Lh</i>	maximum order of rotation coefficients;
<i>wb</i>	parent box width;
<i>work</i>	workspace

**Returns**

0 on success

## 5.9 Indexing and lookup operations

Indexing functions for accessing tree data structures.

### Functions

- quint64 **wbfmm\_box\_index** (quint32 *i*, quint32 *j*, quint32 *k*)
- gint **wbfmm\_box\_location** (quint64 *idx*, quint32 \**i*, quint32 \**j*, quint32 \**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.umi.acs.umd.edu/~gumerov/PDFs/cs-tr-4458.pdf>

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

### 5.9.2 Function Documentation

#### 5.9.2.1 quint64 wbfmm\_box\_index ( quint32 *i*, quint32 *j*, quint32 *k* )

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

##### Parameters

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

##### Returns

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

#### 5.9.2.2 gint wbfmm\_box\_location ( quint64 *idx*, quint32 \**i*, quint32 \**j*, quint32 \**k* )

Compute indices for bottom left hand corner of box defined by its Morton index, as generated by **wbfmm\_box\_index** (p. 70)

##### Parameters

<i>idx</i>	index of box corner;
<i>i</i>	on output, x index of bottom left hand corner of box;
<i>j</i>	on output, y index of bottom left hand corner of box;
<i>k</i>	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

- quint32 **i**
- quint32 **n**
- gpointer **mps**
- gpointer **mpr**

#### 6.1.1 Detailed Description

Data type for octree boxes

#### 6.1.2 Field Documentation

##### 6.1.2.1 quint32 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 quint32 wbfmm\_box\_t::n

number of points in box

### 6.2 wbfmm\_shift\_operators\_t Struct Reference

## Data Fields

- gsize **size**
- quint **nlevels**
- quint **L** [WBFMM\_TREE\_MAX\_DEPTH+1]
- quint **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 quint wbfmm\_shift\_operators\_t::L[WBFMM\_TREE\_MAX\_DEPTH+1]

< number of levels in tree

#### 6.2.2.2 quint wbfmm\_shift\_operators\_t::nerot

< maximum order of expansion per level

#### 6.2.2.3 quint 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\_target\_list\_t Struct Reference

### Data Fields

- **wbfmm\_tree\_t** \* t

- quint **maxpoints**
- quint **npoints**
- quint \* **ip**
- quint **nc**
- quint32 \* **boxes**
- gchar \* **points**
- gsize **size**
- gsize **pstr**
- gint \* **ibox**
- gint \* **isrc**
- gint \* **ics**
- gpointer **cfft**
- gpointer **csrc**
- gboolean **grad**

### 6.3.1 Detailed Description

Data type for target point lists

### 6.3.2 Field Documentation

#### 6.3.2.1 quint32\* wbfmm\_target\_list\_t::boxes

box indices of points

#### 6.3.2.2 gpointer wbfmm\_target\_list\_t::cfft

coefficients of regular expansions in boxes

#### 6.3.2.3 gpointer wbfmm\_target\_list\_t::csrc

coefficients of near-field (direct) interactions, point-by-point

#### 6.3.2.4 gboolean wbfmm\_target\_list\_t::grad

gradient computations included

#### 6.3.2.5 gint\* wbfmm\_target\_list\_t::ibox

start and end of source index lists for each box

#### 6.3.2.6 gint \* wbfmm\_target\_list\_t::ics

start of near-field coefficients for each target

#### 6.3.2.7 quint \* wbfmm\_target\_list\_t::ip

indices of points, sorted by Morton index

**6.3.2.8** `gint * wbfmm_target_list_t::isrc`

source index lists for each box

**6.3.2.9** `guint wbfmm_target_list_t::maxpoints`

maximum number of points in target list

**6.3.2.10** `guint wbfmm_target_list_t::nc`

number of coefficients (size of blocks of coefficients)

**6.3.2.11** `guint wbfmm_target_list_t::npoints`

number of points in target list

**6.3.2.12** `gchar* wbfmm_target_list_t::points`

point coordinates

**6.3.2.13** `gsize wbfmm_target_list_t::pstr`

stride in point data

**6.3.2.14** `gsize wbfmm_target_list_t::size`

size of floating point type in data (float, double, etc)

**6.3.2.15** `wbfmm_tree_t* wbfmm_target_list_t::t`

tree containing source data

## 6.4 wbfmm\_tree\_t Struct Reference

### Data Fields

- **wbfmm\_box\_t \* boxes** [WBFMM\_TREE\_MAX\_DEPTH+1]
- **guint maxpoints**
- **guint npoints**
- **guint \* ip**
- **guint nq**
- **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 size**
- **gsize pstr**
- **gdouble D**

### 6.4.1 Detailed Description

Data type for octrees

### 6.4.2 Field Documentation

#### 6.4.2.1 wbfmm\_box\_t\* wbfmm\_tree\_t::boxes[WBFMM\_TREE\_MAX\_DEPTH+1]

arrays of boxes at each level

Referenced by wbfmm\_tree\_add\_level().

#### 6.4.2.2 gdouble wbfmm\_tree\_t::D

width of domain cube

#### 6.4.2.3 guint wbfmm\_tree\_t::depth

depth of tree

Referenced by wbfmm\_tree\_add\_level().

#### 6.4.2.4 guint \* wbfmm\_tree\_t::ip

indices of points, sorted by Morton index

#### 6.4.2.5 guint wbfmm\_tree\_t::maxpoints

maximum number of points in tree

#### 6.4.2.6 gpointer \* wbfmm\_tree\_t::mpr[WBFMM\_TREE\_MAX\_DEPTH+1]

regular expansion data at each level

#### 6.4.2.7 gpointer\* wbfmm\_tree\_t::mps[WBFMM\_TREE\_MAX\_DEPTH+1]

singular expansion data at each level

#### 6.4.2.8 guint wbfmm\_tree\_t::npoints

number of points in tree

#### 6.4.2.9 guint wbfmm\_tree\_t::nq

number of source components

#### 6.4.2.10 guint wbfmm\_tree\_t::order\_r[WBFMM\_TREE\_MAX\_DEPTH+1]

order of regular expansions at each level

6.4.2.11 `guint wbfmm_tree_t::order_s[WBFMM_TREE_MAX_DEPTH+1]`

order of singular expansions at each level

6.4.2.12 `gchar * wbfmm_tree_t::points`

point coordinates

6.4.2.13 `gsize wbfmm_tree_t::pstr`

stride in point data

6.4.2.14 `gsize wbfmm_tree_t::size`

size of floating point type in data (float, double, etc)

6.4.2.15 `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\_target\_list\_t**
- struct **wbfmm\_shift\_operators\_t**

#### Enumerations

- enum **wbfmm\_problem\_t** { **WBFMM\_PROBLEM\_LAPLACE** = 1, **WBFMM\_PROBLEM\_HELMHOLTZ** = 2 }

#### Functions

- gint **wbfmm\_cartesian\_to\_spherical** (gdouble \*x0, gdouble \*x, gdouble \*r, gdouble \*th, gdouble \*ph)  
*Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .*

- 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  $(\theta, \phi, \chi)$  between axes.*
- gint **wbfmm\_coefficients\_H\_rotation** (gdouble \*H, gint N, gdouble th, gdouble \*work)  
*Compute rotation coefficients for angle  $\theta$ .*
- gint **wbfmm\_laplace\_expansion\_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\_laplace\_field** (gdouble \*xs, gint xstride, gdouble \*src, gint sstride, gint nq, gdouble \*normals, gint nstr, gdouble \*dipoles, gint dstr, gint nsrc, gdouble \*xf, gdouble \*field)
- gint **wbfmm\_laplace\_expansion\_local\_evaluate** (gdouble \*x0, gdouble \*cfft, gint cstr, gint N, gint nq, gdouble \*xf, gdouble \*field, gdouble \*work)
- gint **wbfmm\_laplace\_expansion\_local\_evaluate\_f** (gfloat \*x0, gfloat \*cfft, gint cstr, gint N, gint nq, gfloat \*xf, gfloat \*field, gfloat \*work)
- gint **wbfmm\_laplace\_coaxial\_translate\_init** (gint N)  
*Initialize lookup tables of Laplace translation coefficients.*
- gint **wbfmm\_laplace\_coaxial\_translate\_init\_f** (gint N)



*Initialize lookup tables of Laplace translation coefficients.*

- gint **wbfmm\_laplace\_expansion\_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\_laplace\_field\_f** (gfloat \*xs, gint xstride, gfloat \*src, gint sstride, gint nq, gfloat \*normals, gint nstr, gfloat \*dipoles, gint dstr, gint nsrc, gfloat \*xf, gfloat \*field)
- gint **wbfmm\_laplace\_coaxial\_translate\_SS** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)

*Singular to singular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_SS\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Singular to singular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_RR** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)

*Regular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_RR\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Regular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_SR** (gdouble \*Co, gint cstro, gint No, gdouble \*Ci, gint cstri, gint Ni, gint nq, gdouble t)

*Singular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_coaxial\_translate\_SR\_f** (gfloat \*Co, gint cstro, gint No, gfloat \*Ci, gint cstri, gint Ni, gint nq, gfloat t)

*Singular to regular translation for Laplace expansion.*

- gint **wbfmm\_laplace\_rotate\_H** (gdouble \*Co, gint cstro, gdouble \*Ci, gint cstri, gint N, gint nq, gdouble \*H, gdouble ph, gdouble ch)

*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.*

- gint **wbfmm\_laplace\_rotate\_H\_f** (gfloat \*Co, gint cstro, gfloat \*Ci, gint cstri, gint N, gint nq, gfloat \*H, gfloat ph, gfloat ch)

*Apply rotation  $(\theta, \phi, \chi)$  to multipole coefficients for the Laplace problem.*

- gint **wbfmm\_laplace\_child\_parent\_shift** (gdouble \*Cp, gint Np, gdouble \*Cc, gint Nc, gint nq, gdouble \*H03, gdouble \*H47, gint Lh, gdouble t, gdouble \*work)

*Upward shift of singular expansion from eight children to common parent in Laplace problem.*

- gint **wbfmm\_laplace\_child\_parent\_shift\_f** (gfloat \*Cp, gint Np, gfloat \*Cc, gint Nc, gint nq, gfloat \*H03, gfloat \*H47, gint Lh, gfloat t, gfloat \*work)

*Upward shift of singular expansion from eight children to common parent in Laplace problem.*

- gint **wbfmm\_laplace\_parent\_child\_shift** (gdouble \*Cc, gint Nc, gdouble \*Cp, gint Np, gint nq, gdouble \*H03, gdouble \*H47, gint Lh, gdouble t, gdouble \*work)

*Downward shift of regular expansion from parent to eight children in Laplace problem.*

- gint **wbfmm\_laplace\_parent\_child\_shift\_f** (gfloat \*Cc, gint Nc, gfloat \*Cp, gint Np, gint nq, gfloat \*H03, gfloat \*H47, gint Lh, gfloat t, gfloat \*work)

*Downward shift of regular expansion from parent to eight children in Laplace problem.*

- gint **wbfmm\_tree\_laplace\_leaf\_expansions** (**wbfmm\_tree\_t** \*t, gdouble \*src, gint sstr, gdouble \*normals, gint nstr, gdouble \*dipoles, gint dstr, gboolean zero\_expansions, gdouble \*work)

*Generate leaf expansions for a tree in the Laplace problem.*

- gint **wbfmm\_tree\_laplace\_leaf\_expansions\_f** (**wbfmm\_tree\_t** \*t, gfloat \*src, gint sstr, gfloat \*normals, gint nstr, gfloat \*dipoles, gint dstr, gboolean zero\_expansions, gfloat \*work)

*Generate leaf expansions for a tree in the Laplace problem.*

- gint **wbfmm\_laplace\_downward\_pass** (**wbfmm\_tree\_t** \*t, **wbfmm\_shift\_operators\_t** \*op, quint level, gdouble \*work)

*Perform downward pass at one level of an octree for the Laplace problem.*

- gint **wbfmm\_laplace\_downward\_pass\_f** (**wbfmm\_tree\_t** \*t, **wbfmm\_shift\_operators\_t** \*op, quint level, gfloat \*work)

*Perform downward pass at one level of an octree for the Laplace problem.*

- gint **wbfmm\_laplace\_upward\_pass** (wbfmm\_tree\_t \*t, wbfmm\_shift\_operators\_t \*op, quint level, gdouble \*work)

*Perform upward pass at one level of an octree for the Laplace problem.*

- gint **wbfmm\_laplace\_upward\_pass\_f** (wbfmm\_tree\_t \*t, wbfmm\_shift\_operators\_t \*op, quint level, gfloat \*work)

*Perform upward pass at one level of an octree for the Laplace problem.*

- gint **wbfmm\_tree\_laplace\_box\_local\_field** (wbfmm\_tree\_t \*t, quint level, quint b, gdouble \*x, gdouble \*f, gdouble \*src, gintsstr, gdouble \*normals, gint nstr, gdouble \*d, gint dstr, gboolean eval\_neighbours, gdouble \*work)

*Evaluate local Laplace field from regular expansion in box.*

- gint **wbfmm\_tree\_laplace\_box\_local\_field\_f** (wbfmm\_tree\_t \*t, quint level, quint b, gfloat \*x, gfloat \*f, gfloat \*src, gintsstr, gfloat \*normals, gint nstr, gfloat \*d, gint dstr, gboolean eval\_neighbours, gfloat \*work)

*Evaluate local Laplace field from regular expansion in box.*

- gint **wbfmm\_laplace\_local\_coefficients** (gdouble \*x, gint N, gboolean grad, gdouble \*cfft, gdouble \*work)

*Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)*

- gint **wbfmm\_laplace\_local\_coefficients\_f** (gfloat \*x, gint N, gboolean grad, gfloat \*cfft, gfloat \*work)

*Generate coefficients for evaluation of local field from (regular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)*

- gint **wbfmm\_laplace\_field\_coefficients** (gdouble \*x, gint N, gboolean grad, gdouble \*cfft, gdouble \*work)

*Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply** (p. 65)(...)*

- gint **wbfmm\_laplace\_field\_coefficients\_f** (gfloat \*x, gint N, gboolean grad, gfloat \*cfft, gfloat \*work)

*Generate coefficients for evaluation of field from (singular) expansion coefficients in the Laplace problem. The coefficients from this function can be applied to an expansion using **wbfmm\_laplace\_expansion\_apply\_f** (p. 65)(...)*

- gint **wbfmm\_laplace\_expansion\_apply** (gdouble \*C, gint cstr, gint nq, gdouble \*ec, gint N, gdouble \*f)

*Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using **wbfmm\_laplace\_field\_coefficients** or **wbfmm\_laplace\_local\_coefficients** for the field (singular) or local (regular) expansions respectively.*

- gint **wbfmm\_laplace\_expansion\_apply\_f** (gfloat \*C, gint cstr, gint nq, gfloat \*ec, gint N, gfloat \*f)

*Apply evaluation coefficients to coefficients of an expansion to evaluate the Laplace potential. Evaluation coefficients can be evaluated using **wbfmm\_laplace\_field\_coefficients** or **wbfmm\_laplace\_local\_coefficients** for the field (singular) or local (regular) expansions respectively.*

- 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** (quint L, gdouble \*work)

*Allocate shift operators and initialize rotations.*

- gint **wbfmm\_shift\_operators\_coaxial\_SR\_init** (wbfmm\_shift\_operators\_t \*w, gdouble D, quint level, quint L, gdouble k, gdouble \*work)

*Initialize singular-to-regular translation operators.*

- gint **wbfmm\_shift\_operators\_coaxial\_SS\_init** (wbfmm\_shift\_operators\_t \*w, gdouble D, quint level, quint L, gdouble k, gdouble \*work)

*Initialize singular-to-singular (regular-to-regular) translation operators.*

- gint **wbfmm\_upward\_pass** (wbfmm\_tree\_t \*t, wbfmm\_shift\_operators\_t \*op, quint 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, quint level, gdouble \*work)
- Perform downward pass at one level of an octree.*
- gint **wbfmm\_tree\_box\_field** (wbfmm\_tree\_t \*t, quint level, quint b, gdouble k, gdouble \*x, gdouble \*f, gdouble \*work)
- Evaluate singular expansion about a box centre.*
- 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, quint 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, quint maxpoints)
- Allocate a new octree.*
- gint **wbfmm\_tree\_coefficient\_init** (wbfmm\_tree\_t \*t, quint l, quint nr, quint ns)
- Initialize expansion coefficient data in an octree.*
- gint **wbfmm\_tree\_leaf\_expansions** (wbfmm\_tree\_t \*t, gdouble k, gdouble \*src, gintsstr, gdouble \*normals, gintsstr, gdouble \*dipoles, gintsstr, gboolean zero\_expansions, gdouble \*work)
- Generate leaf expansions for a tree.*
- gint **wbfmm\_box\_location\_from\_index** (guint64 i, quint32 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\_cartesian\_to\_spherical\_f** (gfloat \*x0, gfloat \*x, gfloat \*r, gfloat \*th, gfloat \*ph)
- Convert Cartesian to spherical coordinates  $(r, \theta, \phi)$ .*
- 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 ( $\theta, \phi, \chi$ ) between axes.*

- gint **wbfmm\_coefficients\_H\_rotation\_f** (gfloat \*H, gint N, gfloat th, gfloat \*work)

*Compute rotation coefficients for angle  $\theta$ .*

- 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.*
- gint **wbfmm\_tree\_box\_field\_f** (wbfmm\_tree\_t \*t, guint level, guint b, gfloat k, gfloat \*x, gfloat \*f, gfloat \*work)  
*Evaluate singular expansion about a box centre.*
- 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](mailto:ensmjc@rpc-ensmjc.bath.ac.uk)

#### Date

Mon Jun 24 10:28:46 2019



# Index

## boxes

- wbfmm\_target\_list\_t, 73
- wbfmm\_tree\_t, 75

## Boxes and octrees, 9

- WBFMM\_PROBLEM\_HELMHOLTZ, 10
- WBFMM\_PROBLEM\_LAPLACE, 10
- wbfmm\_point\_index\_3d, 11
- wbfmm\_point\_index\_3d\_f, 12
- wbfmm\_problem\_t, 10
- wbfmm\_tree\_add\_level, 12
- wbfmm\_tree\_add\_points, 12
- wbfmm\_tree\_add\_points\_f, 13
- wbfmm\_tree\_box\_field, 13
- wbfmm\_tree\_box\_field\_f, 13
- wbfmm\_tree\_box\_local\_field, 14
- wbfmm\_tree\_box\_local\_field\_f, 14
- wbfmm\_tree\_coefficient\_init, 14
- wbfmm\_tree\_coefficient\_init\_f, 15
- wbfmm\_tree\_laplace\_box\_local\_field, 15
- wbfmm\_tree\_laplace\_box\_local\_field\_f, 15
- wbfmm\_tree\_laplace\_leaf\_expansions, 16
- wbfmm\_tree\_laplace\_leaf\_expansions\_f, 16
- wbfmm\_tree\_leaf\_expansions, 17
- wbfmm\_tree\_leaf\_expansions\_f, 18
- wbfmm\_tree\_new, 18
- wbfmm\_tree\_new\_f, 19
- wbfmm\_tree\_refine, 19
- wbfmm\_tree\_refine\_f, 19

## cfft

- wbfmm\_target\_list\_t, 73

## csrc

- wbfmm\_target\_list\_t, 73

## D

- wbfmm\_tree\_t, 75

## depth

- wbfmm\_tree\_t, 75

## Evaluation of the Laplace potential, 57

- wbfmm\_box\_fields\_laplace, 58
- wbfmm\_box\_fields\_laplace\_f, 59
- wbfmm\_expansion\_laplace\_evaluate, 59
- wbfmm\_expansion\_laplace\_evaluate\_f, 59
- wbfmm\_laplace\_child\_parent\_shift, 60
- wbfmm\_laplace\_child\_parent\_shift\_f, 60
- wbfmm\_laplace\_coaxial\_translate\_RR, 61
- wbfmm\_laplace\_coaxial\_translate\_RR\_f, 62
- wbfmm\_laplace\_coaxial\_translate\_SR, 62
- wbfmm\_laplace\_coaxial\_translate\_SR\_f, 63

- wbfmm\_laplace\_coaxial\_translate\_SS, 63
- wbfmm\_laplace\_coaxial\_translate\_SS\_f, 63
- wbfmm\_laplace\_coaxial\_translate\_init, 61
- wbfmm\_laplace\_coaxial\_translate\_init\_f, 61
- wbfmm\_laplace\_expansion\_apply, 64
- wbfmm\_laplace\_expansion\_apply\_f, 64
- wbfmm\_laplace\_expansion\_cfft, 65
- wbfmm\_laplace\_expansion\_cfft\_f, 65
- wbfmm\_laplace\_expansion\_local\_evaluate, 65
- wbfmm\_laplace\_expansion\_local\_evaluate\_f, 66
- wbfmm\_laplace\_field, 66
- wbfmm\_laplace\_field\_coefficients, 66
- wbfmm\_laplace\_field\_coefficients\_f, 66
- wbfmm\_laplace\_field\_f, 66
- wbfmm\_laplace\_local\_coefficients, 67
- wbfmm\_laplace\_local\_coefficients\_f, 67
- wbfmm\_laplace\_parent\_child\_shift, 67
- wbfmm\_laplace\_parent\_child\_shift\_f, 68

## Generation and evaluation of expansions, 28

- wbfmm\_expansion\_dipole\_h\_cfft, 28
- wbfmm\_expansion\_dipole\_h\_cfft\_f, 29
- wbfmm\_expansion\_h\_cfft, 29
- wbfmm\_expansion\_h\_cfft\_f, 30
- wbfmm\_expansion\_h\_evaluate, 30
- wbfmm\_expansion\_h\_evaluate\_f, 30
- wbfmm\_expansion\_j\_evaluate, 31
- wbfmm\_expansion\_j\_evaluate\_f, 31

## grad

- wbfmm\_target\_list\_t, 73

## i

- wbfmm\_box\_t, 71

## ibox

- wbfmm\_target\_list\_t, 73

## ics

- wbfmm\_target\_list\_t, 73

## Indexing and lookup operations, 69

- wbfmm\_box\_index, 69
- wbfmm\_box\_location, 69

## ip

- wbfmm\_target\_list\_t, 73
- wbfmm\_tree\_t, 75

## isrc

- wbfmm\_target\_list\_t, 73

## L

- wbfmm\_shift\_operators\_t, 72

## maxpoints

- wbfmm\_target\_list\_t, 74
- wbfmm\_tree\_t, 75
- mpr
  - wbfmm\_box\_t, 71
  - wbfmm\_tree\_t, 75
- mps
  - wbfmm\_box\_t, 71
  - wbfmm\_tree\_t, 75
- n
  - wbfmm\_box\_t, 71
- nc
  - wbfmm\_target\_list\_t, 74
- nerot
  - wbfmm\_shift\_operators\_t, 72
- nlevels
  - wbfmm\_shift\_operators\_t, 72
- npoints
  - wbfmm\_target\_list\_t, 74
  - wbfmm\_tree\_t, 75
- nq
  - wbfmm\_tree\_t, 75
- order\_r
  - wbfmm\_tree\_t, 75
- order\_s
  - wbfmm\_tree\_t, 75
- points
  - wbfmm\_target\_list\_t, 74
  - wbfmm\_tree\_t, 76
- pstr
  - wbfmm\_target\_list\_t, 74
  - wbfmm\_tree\_t, 76
- Rotation coefficients and operations, 36
  - wbfmm\_coefficients\_H\_rotation, 36
  - wbfmm\_coefficients\_H\_rotation\_f, 37
  - wbfmm\_laplace\_rotate\_H, 37
  - wbfmm\_laplace\_rotate\_H\_f, 37
  - wbfmm\_rotate\_H, 38
  - wbfmm\_rotate\_H\_f, 38
  - wbfmm\_rotation\_angles, 39
  - wbfmm\_rotation\_angles\_f, 39
- rotations
  - wbfmm\_shift\_operators\_t, 72
- SR
  - wbfmm\_shift\_operators\_t, 72
- SS
  - wbfmm\_shift\_operators\_t, 72
- Shift operations, 20
  - wbfmm\_child\_parent\_shift, 21
  - wbfmm\_child\_parent\_shift\_bw, 21
  - wbfmm\_child\_parent\_shift\_bw\_f, 22
  - wbfmm\_child\_parent\_shift\_f, 22
  - wbfmm\_parent\_child\_shift, 23
  - wbfmm\_parent\_child\_shift\_f, 23
  - wbfmm\_shift\_angle\_table\_init, 24
  - wbfmm\_shift\_angle\_table\_init\_f, 24
  - wbfmm\_shift\_angles\_list4, 24
  - wbfmm\_shift\_angles\_list4\_f, 25
  - wbfmm\_shift\_operators\_coaxial\_SR\_init, 25
  - wbfmm\_shift\_operators\_coaxial\_SS\_init, 26
  - wbfmm\_shift\_operators\_new, 26
  - wbfmm\_shift\_operators\_new\_f, 27
- size
  - wbfmm\_shift\_operators\_t, 72
  - wbfmm\_target\_list\_t, 74
  - wbfmm\_tree\_t, 76
- t
  - wbfmm\_target\_list\_t, 74
- Translation operators, 41
  - wbfmm\_coaxial\_translate, 41
  - wbfmm\_coaxial\_translate\_f, 42
  - wbfmm\_coefficients\_RR\_coaxial, 42
  - wbfmm\_coefficients\_RR\_coaxial\_f, 25
  - wbfmm\_coefficients\_SR\_coaxial, 43
  - wbfmm\_coefficients\_SR\_coaxial\_f, 43
- tree.c, 77
- Upward and downward passes, 32
  - wbfmm\_downward\_pass, 32
  - wbfmm\_downward\_pass\_f, 33
  - wbfmm\_laplace\_downward\_pass, 33
  - wbfmm\_laplace\_downward\_pass\_f, 33
  - wbfmm\_laplace\_upward\_pass, 34
  - wbfmm\_laplace\_upward\_pass\_f, 34
  - wbfmm\_upward\_pass, 34
  - wbfmm\_upward\_pass\_f, 35
- Utility and convenience functions, 44
  - wbfmm\_bessel\_h\_init, 45
  - wbfmm\_bessel\_h\_init\_f, 45
  - wbfmm\_bessel\_h\_recursion, 46
  - wbfmm\_bessel\_h\_recursion\_f, 46
  - wbfmm\_bessel\_j\_init, 46
  - wbfmm\_bessel\_j\_init\_f, 47
  - wbfmm\_bessel\_j\_recursion, 47
  - wbfmm\_bessel\_j\_recursion\_f, 47
  - wbfmm\_box\_location\_from\_index, 48
  - wbfmm\_box\_location\_from\_index\_f, 48
  - wbfmm\_cartesian\_to\_spherical, 48
  - wbfmm\_cartesian\_to\_spherical\_f, 49
  - wbfmm\_coordinate\_transform, 49
  - wbfmm\_coordinate\_transform\_f, 49
  - wbfmm\_legendre\_init, 50
  - wbfmm\_legendre\_init\_f, 50
  - wbfmm\_legendre\_recursion\_array, 50
  - wbfmm\_legendre\_recursion\_array\_f, 51
  - wbfmm\_points\_origin\_width, 51
  - wbfmm\_points\_origin\_width\_f, 51
  - wbfmm\_shift\_angles, 52
  - wbfmm\_shift\_angles\_f, 52
  - wbfmm\_shift\_coordinates, 52
  - wbfmm\_shift\_coordinates\_f, 53
  - wbfmm\_total\_dipole\_field, 53
  - wbfmm\_total\_dipole\_field\_f, 53



- wbfmm\_total\_field, 54
- wbfmm\_total\_field\_f, 54
- wbfmm\_tree\_box\_centre, 55
- wbfmm\_tree\_box\_centre\_f, 55
- wbfmm\_tree\_write\_sources, 55
- wbfmm\_tree\_write\_sources\_f, 55
- WBFMM\_PROBLEM\_HELMHOLTZ
  - Boxes and octrees, 10
- WBFMM\_PROBLEM\_LAPLACE
  - Boxes and octrees, 10
- wbfmm.h, 77
- wbfmm\_bessel\_h\_init
  - Utility and convenience functions, 45
- wbfmm\_bessel\_h\_init\_f
  - Utility and convenience functions, 45
- wbfmm\_bessel\_h\_recursion
  - Utility and convenience functions, 46
- wbfmm\_bessel\_h\_recursion\_f
  - Utility and convenience functions, 46
- wbfmm\_bessel\_j\_init
  - Utility and convenience functions, 46
- wbfmm\_bessel\_j\_init\_f
  - Utility and convenience functions, 47
- wbfmm\_bessel\_j\_recursion
  - Utility and convenience functions, 47
- wbfmm\_bessel\_j\_recursion\_f
  - Utility and convenience functions, 47
- wbfmm\_box\_fields\_laplace
  - Evaluation of the Laplace potential, 58
- wbfmm\_box\_fields\_laplace\_f
  - Evaluation of the Laplace potential, 59
- wbfmm\_box\_index
  - Indexing and lookup operations, 69
- wbfmm\_box\_location
  - Indexing and lookup operations, 69
- wbfmm\_box\_location\_from\_index
  - Utility and convenience functions, 48
- wbfmm\_box\_location\_from\_index\_f
  - Utility and convenience functions, 48
- wbfmm\_box\_t, 71
  - i, 71
  - mpr, 71
  - mps, 71
  - n, 71
- wbfmm\_cartesian\_to\_spherical
  - Utility and convenience functions, 48
- wbfmm\_cartesian\_to\_spherical\_f
  - Utility and convenience functions, 49
- wbfmm\_child\_parent\_shift
  - Shift operations, 21
- wbfmm\_child\_parent\_shift\_bw
  - Shift operations, 21
- wbfmm\_child\_parent\_shift\_bw\_f
  - Shift operations, 22
- wbfmm\_child\_parent\_shift\_f
  - Shift operations, 22
- wbfmm\_coaxial\_translate
  - Translation operators, 41
- wbfmm\_coaxial\_translate\_f
  - Translation operators, 42
- wbfmm\_coefficients\_H\_rotation
  - Rotation coefficients and operations, 36
- wbfmm\_coefficients\_H\_rotation\_f
  - Rotation coefficients and operations, 37
- wbfmm\_coefficients\_RR\_coaxial
  - Translation operators, 42
- wbfmm\_coefficients\_RR\_coaxial\_f
  - Translation operators, 42
- wbfmm\_coefficients\_SR\_coaxial
  - Translation operators, 43
- wbfmm\_coefficients\_SR\_coaxial\_f
  - Translation operators, 43
- wbfmm\_coordinate\_transform
  - Utility and convenience functions, 49
- wbfmm\_coordinate\_transform\_f
  - Utility and convenience functions, 49
- wbfmm\_downward\_pass
  - Upward and downward passes, 32
- wbfmm\_downward\_pass\_f
  - Upward and downward passes, 33
- wbfmm\_expansion\_dipole\_h\_cfft
  - Generation and evaluation of expansions, 28
- wbfmm\_expansion\_dipole\_h\_cfft\_f
  - Generation and evaluation of expansions, 29
- wbfmm\_expansion\_h\_cfft
  - Generation and evaluation of expansions, 29
- wbfmm\_expansion\_h\_cfft\_f
  - Generation and evaluation of expansions, 30
- wbfmm\_expansion\_h\_evaluate
  - Generation and evaluation of expansions, 30
- wbfmm\_expansion\_h\_evaluate\_f
  - Generation and evaluation of expansions, 30
- wbfmm\_expansion\_j\_evaluate
  - Generation and evaluation of expansions, 31
- wbfmm\_expansion\_j\_evaluate\_f
  - Generation and evaluation of expansions, 31
- wbfmm\_expansion\_laplace\_evaluate
  - Evaluation of the Laplace potential, 59
- wbfmm\_expansion\_laplace\_evaluate\_f
  - Evaluation of the Laplace potential, 59
- wbfmm\_laplace\_child\_parent\_shift
  - Evaluation of the Laplace potential, 60
- wbfmm\_laplace\_child\_parent\_shift\_f
  - Evaluation of the Laplace potential, 60
- wbfmm\_laplace\_coaxial\_translate\_RR
  - Evaluation of the Laplace potential, 61
- wbfmm\_laplace\_coaxial\_translate\_RR\_f
  - Evaluation of the Laplace potential, 62
- wbfmm\_laplace\_coaxial\_translate\_SR
  - Evaluation of the Laplace potential, 62
- wbfmm\_laplace\_coaxial\_translate\_SR\_f
  - Evaluation of the Laplace potential, 63
- wbfmm\_laplace\_coaxial\_translate\_SS
  - Evaluation of the Laplace potential, 63
- wbfmm\_laplace\_coaxial\_translate\_SS\_f
  - Evaluation of the Laplace potential, 63

- wbfmm\_laplace\_coaxial\_translate\_init
  - Evaluation of the Laplace potential, 61
- wbfmm\_laplace\_coaxial\_translate\_init\_f
  - Evaluation of the Laplace potential, 61
- wbfmm\_laplace\_downward\_pass
  - Upward and downward passes, 33
- wbfmm\_laplace\_downward\_pass\_f
  - Upward and downward passes, 33
- wbfmm\_laplace\_expansion\_apply
  - Evaluation of the Laplace potential, 64
- wbfmm\_laplace\_expansion\_apply\_f
  - Evaluation of the Laplace potential, 64
- wbfmm\_laplace\_expansion\_cfft
  - Evaluation of the Laplace potential, 65
- wbfmm\_laplace\_expansion\_cfft\_f
  - Evaluation of the Laplace potential, 65
- wbfmm\_laplace\_expansion\_local\_evaluate
  - Evaluation of the Laplace potential, 65
- wbfmm\_laplace\_expansion\_local\_evaluate\_f
  - Evaluation of the Laplace potential, 66
- wbfmm\_laplace\_field
  - Evaluation of the Laplace potential, 66
- wbfmm\_laplace\_field\_coefficients
  - Evaluation of the Laplace potential, 66
- wbfmm\_laplace\_field\_coefficients\_f
  - Evaluation of the Laplace potential, 66
- wbfmm\_laplace\_field\_f
  - Evaluation of the Laplace potential, 66
- wbfmm\_laplace\_local\_coefficients
  - Evaluation of the Laplace potential, 67
- wbfmm\_laplace\_local\_coefficients\_f
  - Evaluation of the Laplace potential, 67
- wbfmm\_laplace\_parent\_child\_shift
  - Evaluation of the Laplace potential, 67
- wbfmm\_laplace\_parent\_child\_shift\_f
  - Evaluation of the Laplace potential, 68
- wbfmm\_laplace\_rotate\_H
  - Rotation coefficients and operations, 37
- wbfmm\_laplace\_rotate\_H\_f
  - Rotation coefficients and operations, 37
- wbfmm\_laplace\_upward\_pass
  - Upward and downward passes, 34
- wbfmm\_laplace\_upward\_pass\_f
  - Upward and downward passes, 34
- wbfmm\_legendre\_init
  - Utility and convenience functions, 50
- wbfmm\_legendre\_init\_f
  - Utility and convenience functions, 50
- wbfmm\_legendre\_recursion\_array
  - Utility and convenience functions, 50
- wbfmm\_legendre\_recursion\_array\_f
  - Utility and convenience functions, 51
- wbfmm\_parent\_child\_shift
  - Shift operations, 23
- wbfmm\_parent\_child\_shift\_f
  - Shift operations, 23
- wbfmm\_point\_index\_3d
  - Boxes and octrees, 11
- wbfmm\_point\_index\_3d\_f
  - Boxes and octrees, 12
- wbfmm\_points\_origin\_width
  - Utility and convenience functions, 51
- wbfmm\_points\_origin\_width\_f
  - Utility and convenience functions, 51
- wbfmm\_problem\_t
  - Boxes and octrees, 10
- wbfmm\_rotate\_H
  - Rotation coefficients and operations, 38
- wbfmm\_rotate\_H\_f
  - Rotation coefficients and operations, 38
- wbfmm\_rotation\_angles
  - Rotation coefficients and operations, 39
- wbfmm\_rotation\_angles\_f
  - Rotation coefficients and operations, 39
- wbfmm\_shift\_angle\_table\_init
  - Shift operations, 24
- wbfmm\_shift\_angle\_table\_init\_f
  - Shift operations, 24
- wbfmm\_shift\_angles
  - Utility and convenience functions, 52
- wbfmm\_shift\_angles\_f
  - Utility and convenience functions, 52
- wbfmm\_shift\_angles\_list4
  - Shift operations, 24
- wbfmm\_shift\_angles\_list4\_f
  - Shift operations, 25
- wbfmm\_shift\_coordinates
  - Utility and convenience functions, 52
- wbfmm\_shift\_coordinates\_f
  - Utility and convenience functions, 53
- wbfmm\_shift\_operators\_coaxial\_SR\_init
  - Shift operations, 25
- wbfmm\_shift\_operators\_coaxial\_SS\_init
  - Shift operations, 26
- wbfmm\_shift\_operators\_new
  - Shift operations, 26
- wbfmm\_shift\_operators\_new\_f
  - Shift operations, 27
- wbfmm\_shift\_operators\_t, 71
  - L, 72
  - nerot, 72
  - nlevels, 72
  - rotations, 72
  - SR, 72
  - SS, 72
  - size, 72
- wbfmm\_target\_list\_t, 72
  - boxes, 73
  - cfft, 73
  - csrc, 73
  - grad, 73
  - ibox, 73
  - ics, 73
  - ip, 73
  - isrc, 73
  - maxpoints, 74

- nc, 74
- npoints, 74
- points, 74
- pstr, 74
- size, 74
- t, 74
- wbfmm\_total\_dipole\_field
  - Utility and convenience functions, 53
- wbfmm\_total\_dipole\_field\_f
  - Utility and convenience functions, 53
- wbfmm\_total\_field
  - Utility and convenience functions, 54
- wbfmm\_total\_field\_f
  - Utility and convenience functions, 54
- wbfmm\_tree\_add\_level
  - Boxes and octrees, 12
- wbfmm\_tree\_add\_points
  - Boxes and octrees, 12
- wbfmm\_tree\_add\_points\_f
  - Boxes and octrees, 13
- wbfmm\_tree\_box\_centre
  - Utility and convenience functions, 55
- wbfmm\_tree\_box\_centre\_f
  - Utility and convenience functions, 55
- wbfmm\_tree\_box\_field
  - Boxes and octrees, 13
- wbfmm\_tree\_box\_field\_f
  - Boxes and octrees, 13
- wbfmm\_tree\_box\_local\_field
  - Boxes and octrees, 14
- wbfmm\_tree\_box\_local\_field\_f
  - Boxes and octrees, 14
- wbfmm\_tree\_coefficient\_init
  - Boxes and octrees, 14
- wbfmm\_tree\_coefficient\_init\_f
  - Boxes and octrees, 15
- wbfmm\_tree\_laplace\_box\_local\_field
  - Boxes and octrees, 15
- wbfmm\_tree\_laplace\_box\_local\_field\_f
  - Boxes and octrees, 15
- wbfmm\_tree\_laplace\_leaf\_expansions
  - Boxes and octrees, 16
- wbfmm\_tree\_laplace\_leaf\_expansions\_f
  - Boxes and octrees, 16
- wbfmm\_tree\_leaf\_expansions
  - Boxes and octrees, 17
- wbfmm\_tree\_leaf\_expansions\_f
  - Boxes and octrees, 18
- wbfmm\_tree\_new
  - Boxes and octrees, 18
- wbfmm\_tree\_new\_f
  - Boxes and octrees, 19
- wbfmm\_tree\_refine
  - Boxes and octrees, 19
- wbfmm\_tree\_refine\_f
  - Boxes and octrees, 19
- wbfmm\_tree\_t, 74
  - boxes, 75
  - D, 75
  - depth, 75
  - ip, 75
  - maxpoints, 75
  - mpr, 75
  - mps, 75
  - npoints, 75
  - nq, 75
  - order\_r, 75
  - order\_s, 75
  - points, 76
  - pstr, 76
  - size, 76
  - x, 76
- wbfmm\_tree\_write\_sources
  - Utility and convenience functions, 55
- wbfmm\_tree\_write\_sources\_f
  - Utility and convenience functions, 55
- wbfmm\_upward\_pass
  - Upward and downward passes, 34
- wbfmm\_upward\_pass\_f
  - Upward and downward passes, 35
- x
  - wbfmm\_tree\_t, 76