

LUMA

1.2.0-alpha

Generated by Doxygen 1.8.11

Contents

1	Main Page	1
2	Hierarchical Index	3
2.1	Class Hierarchy	3
3	Class Index	5
3.1	Class List	5
4	File Index	7
4.1	File List	7
5	Class Documentation	9
5.1	BFLBody Class Reference	9
5.1.1	Detailed Description	10
5.1.2	Constructor & Destructor Documentation	10
5.1.2.1	BFLBody(void)	10
5.1.2.2	~BFLBody(void)	10
5.1.2.3	BFLBody(PCpts *_PCpts, GridObj *g_hierarchy)	10
5.1.3	Member Function Documentation	10
5.1.3.1	computeQ(int i, int j, int k, GridObj *g)	10
5.1.3.2	computeQ(int i, int j, GridObj *g)	11
5.1.4	Friends And Related Function Documentation	11
5.1.4.1	GridObj	11
5.1.5	Member Data Documentation	11
5.1.5.1	Q	11

5.2	BFLMarker Class Reference	11
5.2.1	Detailed Description	12
5.2.2	Constructor & Destructor Documentation	12
5.2.2.1	BFLMarker(void)	12
5.2.2.2	~BFLMarker(void)	12
5.2.2.3	BFLMarker(double x, double y, double z)	12
5.2.3	Friends And Related Function Documentation	12
5.2.3.1	BFLBody	13
5.3	Body< MarkerType > Class Template Reference	13
5.3.1	Detailed Description	14
5.3.2	Constructor & Destructor Documentation	14
5.3.2.1	Body(void)	14
5.3.2.2	~Body(void)	14
5.3.2.3	Body(GridObj *g)	14
5.3.3	Member Function Documentation	14
5.3.3.1	addMarker(double x, double y, double z)	14
5.3.3.2	getMarkerData(double x, double y, double z)	14
5.3.3.3	isInVoxel(double x, double y, double z, int curr_mark)	15
5.3.3.4	isVoxelMarkerVoxel(double x, double y, double z)	15
5.3.3.5	markerAdder(double x, double y, double z, int &curr_mark, std::vector< int > &counter)	15
5.3.4	Member Data Documentation	16
5.3.4.1	_Owner	16
5.3.4.2	closed_surface	16
5.3.4.3	markers	16
5.3.4.4	spacing	16
5.4	MpiManager::buffer_struct Struct Reference	16
5.4.1	Detailed Description	17
5.4.2	Member Data Documentation	17
5.4.2.1	level	17
5.4.2.2	region	17

5.4.2.3	size	17
5.5	GridObj Class Reference	17
5.5.1	Detailed Description	20
5.5.2	Constructor & Destructor Documentation	20
5.5.2.1	GridObj()	20
5.5.2.2	GridObj(int level)	20
5.5.2.3	GridObj(int RegionNumber, GridObj &pGrid)	20
5.5.2.4	GridObj(int level, std::vector< int > local_size, std::vector< std::vector< int > > GlobalLimsInd, std::vector< std::vector< double > > GlobalLimsPos)	20
5.5.2.5	~GridObj()	21
5.5.3	Member Function Documentation	21
5.5.3.1	bc_applyBfl(int i, int j, int k)	21
5.5.3.2	bc_applyBounceBack(int label, int i, int j, int k)	21
5.5.3.3	bc_applyExtrapolation(int label, int i, int j, int k)	21
5.5.3.4	bc_applyNrbc(int i, int j, int k)	22
5.5.3.5	bc_applyRegularised(int label, int i, int j, int k)	22
5.5.3.6	bc_applySpecReflect(int label, int i, int j, int k)	22
5.5.3.7	bc_solidSiteReset()	23
5.5.3.8	io_hdf5(double tval)	23
5.5.3.9	io_lite(double tval, std::string Tag)	23
5.5.3.10	io_probeOutput()	23
5.5.3.11	io_restart(bool IO_flag)	23
5.5.3.12	io_textout(std::string output_tag)	24
5.5.3.13	LBM_addSubGrid(int RegionNumber)	24
5.5.3.14	LBM_boundary(int bc_type_flag)	24
5.5.3.15	LBM_coalesce(int RegionNumber)	24
5.5.3.16	LBM_collide()	24
5.5.3.17	LBM_collide(int i, int j, int k, int v)	25
5.5.3.18	LBM_explode(int RegionNumber)	25
5.5.3.19	LBM_forcegrid(bool reset_flag)	25
5.5.3.20	LBM_init_getInletProfile()	25

5.5.3.21	LBM_initBoundLab()	26
5.5.3.22	LBM_initGrid()	26
5.5.3.23	LBM_initGrid(std::vector< int > local_size, std::vector< std::vector< int > > GlobalLimsInd, std::vector< std::vector< double > > GlobalLimsPos)	26
5.5.3.24	LBM_initRefinedLab(GridObj &pGrid)	26
5.5.3.25	LBM_initRho()	26
5.5.3.26	LBM_initSolidLab()	26
5.5.3.27	LBM_initSubGrid(GridObj &pGrid)	26
5.5.3.28	LBM_initVelocity()	27
5.5.3.29	LBM_kbcCollide(int i, int j, int k, IVector< double > &f_new)	27
5.5.3.30	LBM_macro()	27
5.5.3.31	LBM_macro(int i, int j, int k)	27
5.5.3.32	LBM_multi(bool IBM_flag)	28
5.5.3.33	LBM_stream()	28
5.5.4	Friends And Related Function Documentation	28
5.5.4.1	GridUtils	28
5.5.4.2	MpiManager	28
5.5.4.3	ObjectManager	28
5.5.5	Member Data Documentation	28
5.5.5.1	dt	28
5.5.5.2	K_lim	28
5.5.5.3	LatTyp	28
5.5.5.4	level	28
5.5.5.5	M_lim	29
5.5.5.6	N_lim	29
5.5.5.7	nu	29
5.5.5.8	omega	29
5.5.5.9	t	29
5.5.5.10	timeav_mpi_overhead	29
5.5.5.11	timeav_timestep	29
5.5.5.12	XInd	29

5.5.5.13	XPos	29
5.5.5.14	YInd	29
5.5.5.15	YPos	30
5.5.5.16	ZInd	30
5.5.5.17	ZPos	30
5.6	GridUtils Class Reference	30
5.6.1	Detailed Description	32
5.6.2	Member Function Documentation	32
5.6.2.1	add(std::vector< double > a, std::vector< double > b)	32
5.6.2.2	createOutputDirectory(std::string path_str)	32
5.6.2.3	crossprod(std::vector< double > vec1, std::vector< double > vec2)	32
5.6.2.4	dotprod(std::vector< double > vec1, std::vector< double > vec2)	32
5.6.2.5	downToLimit(NumType x, NumType limit)	32
5.6.2.6	factorial(NumType n)	32
5.6.2.7	getCoarseIndices(int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)	33
5.6.2.8	getFineIndices(int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)	33
5.6.2.9	getGrid(GridObj *&Grids, int level, int region, GridObj *&ptr)	33
5.6.2.10	getOpposite(int direction)	33
5.6.2.11	getVoxInd(double x, double y, double z)	33
5.6.2.12	getVoxInd(double p)	33
5.6.2.13	global_to_local(int i, int j, int k, GridObj *g, std::vector< NumType > &locals)	33
5.6.2.14	hasThisSubGrid(const GridObj &pGrid, int RegNum)	34
5.6.2.15	indexToPosition(int index, double dx)	34
5.6.2.16	isOffGrid(int i, int j, int k, GridObj &g)	34
5.6.2.17	isOnRecvLayer(double pos_x, double pos_y, double pos_z)	34
5.6.2.18	isOnRecvLayer(double site_position, enum eCartesianDirection xyz, enum e↔ MinMax minmax)	34
5.6.2.19	isOnSenderLayer(double pos_x, double pos_y, double pos_z)	34
5.6.2.20	isOnSenderLayer(double site_position, enum eCartesianDirection xyz, enum e↔ MinMax minmax)	34
5.6.2.21	isOnThisRank(int gi, int gj, int gk, const GridObj &pGrid)	34

5.6.2.22	<code>isOnThisRank(int gl, enum eCartesianDirection xyz, const GridObj &pGrid)</code>	34
5.6.2.23	<code>isOverlapPeriodic(int i, int j, int k, const GridObj &pGrid)</code>	34
5.6.2.24	<code>linspace(double min, double max, int n)</code>	34
5.6.2.25	<code>local_to_global(int i, int j, int k, GridObj *g, std::vector< NumType > &globals)</code>	34
5.6.2.26	<code>matrix_multiply(const std::vector< std::vector< double > > &A, const std::vector< double > &x)</code>	35
5.6.2.27	<code>onespace(int min, int max)</code>	35
5.6.2.28	<code>stridedCopy(NumType *dest, NumType *src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset=0)</code>	35
5.6.2.29	<code>subtract(std::vector< double > a, std::vector< double > b)</code>	35
5.6.2.30	<code>upToZero(NumType x)</code>	35
5.6.2.31	<code>vecmultiply(double scalar, std::vector< double > vec)</code>	35
5.6.2.32	<code>vecnorm(double vec[])</code>	35
5.6.2.33	<code>vecnorm(double val1, double val2)</code>	36
5.6.2.34	<code>vecnorm(double val1, double val2, double val3)</code>	36
5.6.2.35	<code>vecnorm(std::vector< double > vec)</code>	36
5.6.2.36	<code>vecnorm(NumType a1, NumType a2, NumType a3)</code>	36
5.6.2.37	<code>vecnorm(NumType a1, NumType a2)</code>	36
5.6.3	Member Data Documentation	36
5.6.3.1	<code>dir_reflect</code>	36
5.6.3.2	<code>logfile</code>	37
5.6.3.3	<code>path_str</code>	37
5.7	IBBody Class Reference	37
5.7.1	Detailed Description	38
5.7.2	Constructor & Destructor Documentation	38
5.7.2.1	<code>IBBody(void)</code>	38
5.7.2.2	<code>~IBBody(void)</code>	38
5.7.2.3	<code>IBBody(GridObj *g)</code>	38
5.7.3	Member Function Documentation	39
5.7.3.1	<code>addMarker(double x, double y, double z, bool flex_rigid)</code>	39
5.7.3.2	<code>makeBody(double radius, std::vector< double > centre, bool flex_rigid, bool moving, int group)</code>	39

5.7.3.3	<code>makeBody(std::vector< double > width_length_depth, std::vector< double > angles, std::vector< double > centre, bool flex_rigid, bool deform, int group)</code>	39
5.7.3.4	<code>makeBody(int numbermarkers, std::vector< double > start_point, double fil_↵length, std::vector< double > angles, std::vector< int > BCs, bool flex_rigid, bool deform, int group)</code>	40
5.7.3.5	<code>makeBody(std::vector< double > width_length, double angle, std::vector< double > centre, bool flex_rigid, bool deform, int group, bool plate)</code>	40
5.7.3.6	<code>makeBody(PCpts *_PCpts)</code>	40
5.7.4	Friends And Related Function Documentation	40
5.7.4.1	<code>ObjectManager</code>	40
5.7.5	Member Data Documentation	41
5.7.5.1	<code>BCs</code>	41
5.7.5.2	<code>deformable</code>	41
5.7.5.3	<code>delta_rho</code>	41
5.7.5.4	<code>flex_rigid</code>	41
5.7.5.5	<code>flexural_rigidity</code>	41
5.7.5.6	<code>groupID</code>	41
5.7.5.7	<code>tension</code>	41
5.8	IBMarker Class Reference	41
5.8.1	Detailed Description	42
5.8.2	Constructor & Destructor Documentation	42
5.8.2.1	<code>IBMarker(void)</code>	42
5.8.2.2	<code>~IBMarker(void)</code>	43
5.8.2.3	<code>IBMarker(double xPos, double yPos, double zPos, bool flex_rigid=false)</code>	43
5.8.3	Friends And Related Function Documentation	43
5.8.3.1	<code>IBBody</code>	43
5.8.3.2	<code>ObjectManager</code>	43
5.8.4	Member Data Documentation	43
5.8.4.1	<code>deltaval</code>	43
5.8.4.2	<code>desired_vel</code>	43
5.8.4.3	<code>dilation</code>	43
5.8.4.4	<code>epsilon</code>	43

5.8.4.5	flex_rigid	43
5.8.4.6	fluid_vel	44
5.8.4.7	force_xyz	44
5.8.4.8	local_area	44
5.8.4.9	position_old	44
5.9	IVector< GenTyp > Class Template Reference	44
5.9.1	Detailed Description	45
5.9.2	Constructor & Destructor Documentation	45
5.9.2.1	IVector()	45
5.9.2.2	~IVector()	45
5.9.2.3	IVector(size_t size, GenTyp val)	45
5.9.3	Member Function Documentation	45
5.9.3.1	operator()(size_t i, size_t j, size_t k, size_t v, size_t j_max, size_t k_max, size_t v_max)	45
5.9.3.2	operator()(size_t i, size_t j, size_t k, size_t j_max, size_t k_max)	46
5.9.3.3	operator()(size_t i, size_t j, size_t j_max)	46
5.10	MpiManager::layer_edges Struct Reference	46
5.10.1	Detailed Description	47
5.10.2	Member Data Documentation	47
5.10.2.1	X	47
5.10.2.2	Y	47
5.10.2.3	Z	47
5.11	Marker Class Reference	47
5.11.1	Detailed Description	48
5.11.2	Constructor & Destructor Documentation	48
5.11.2.1	Marker(void)	48
5.11.2.2	~Marker(void)	48
5.11.2.3	Marker(double x, double y, double z)	48
5.11.3	Member Data Documentation	49
5.11.3.1	position	49
5.11.3.2	supp_i	49

5.11.3.3	supp_j	49
5.11.3.4	supp_k	49
5.11.3.5	support_rank	49
5.12	MarkerData Class Reference	49
5.12.1	Detailed Description	50
5.12.2	Constructor & Destructor Documentation	50
5.12.2.1	MarkerData(int i, int j, int k, double x, double y, double z, int ID)	50
5.12.2.2	MarkerData(void)	50
5.12.2.3	~MarkerData(void)	50
5.12.3	Member Data Documentation	51
5.12.3.1	i	51
5.12.3.2	ID	51
5.12.3.3	j	51
5.12.3.4	k	51
5.12.3.5	x	51
5.12.3.6	y	51
5.12.3.7	z	51
5.13	MpiManager Class Reference	51
5.13.1	Detailed Description	54
5.13.2	Member Function Documentation	54
5.13.2.1	destroyInstance()	54
5.13.2.2	getInstance()	54
5.13.2.3	mpi_buffer_pack(int dir, GridObj *g)	54
5.13.2.4	mpi_buffer_size()	54
5.13.2.5	mpi_buffer_size_recv(GridObj *g)	54
5.13.2.6	mpi_buffer_size_send(GridObj *g)	55
5.13.2.7	mpi_buffer_unpack(int dir, GridObj *g)	55
5.13.2.8	mpi_buildCommunicators()	55
5.13.2.9	mpi_communicate(int level, int regnum)	55
5.13.2.10	mpi_getOpposite(int direction)	56

5.13.2.11 <code>mpi_gridbuild()</code>	56
5.13.2.12 <code>mpi_init()</code>	56
5.13.2.13 <code>mpi_writeout_buf(std::string filename, int dir)</code>	56
5.13.3 Member Data Documentation	56
5.13.3.1 <code>buffer_recv_info</code>	56
5.13.3.2 <code>buffer_send_info</code>	56
5.13.3.3 <code>f_buffer_recv</code>	56
5.13.3.4 <code>f_buffer_send</code>	57
5.13.3.5 <code>global_dims</code>	57
5.13.3.6 <code>global_edge_ind</code>	57
5.13.3.7 <code>global_edge_pos</code>	57
5.13.3.8 <code>Grids</code>	57
5.13.3.9 <code>local_size</code>	57
5.13.3.10 <code>logout</code>	57
5.13.3.11 <code>MPI_cartlab</code>	57
5.13.3.12 <code>MPI_coords</code>	58
5.13.3.13 <code>MPI_dims</code>	58
5.13.3.14 <code>my_rank</code>	58
5.13.3.15 <code>neighbour_coords</code>	58
5.13.3.16 <code>neighbour_rank</code>	58
5.13.3.17 <code>num_ranks</code>	58
5.13.3.18 <code>p_data</code>	58
5.13.3.19 <code>recv_layer_pos</code>	58
5.13.3.20 <code>recv_stat</code>	58
5.13.3.21 <code>send_requests</code>	58
5.13.3.22 <code>send_stat</code>	59
5.13.3.23 <code>sender_layer_pos</code>	59
5.13.3.24 <code>subGrid_comm</code>	59
5.13.3.25 <code>world_comm</code>	59
5.14 ObjectManager Class Reference	59

5.14.1 Detailed Description	61
5.14.2 Member Function Documentation	61
5.14.2.1 bfl_build_body(int body_type)	61
5.14.2.2 bfl_build_body(PCpts *_PCpts)	61
5.14.2.3 computeLiftDrag(int i, int j, int k, GridObj *g)	61
5.14.2.4 destroyInstance()	62
5.14.2.5 getInstance()	62
5.14.2.6 getInstance(GridObj *g)	62
5.14.2.7 ibm_apply(GridObj &g)	62
5.14.2.8 ibm_banbks(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double b[])	62
5.14.2.9 ibm_banded(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double *d)	62
5.14.2.10 ibm_bicgstab(std::vector< std::vector< double > > &Amatrix, std::vector< double > &bVector, std::vector< double > &epsilon, double tolerance, int maxiterations)	62
5.14.2.11 ibm_build_body(int body_type)	62
5.14.2.12 ibm_build_body(PCpts *_PCpts, GridObj *owner)	62
5.14.2.13 ibm_computeforce(int ib, GridObj &g)	63
5.14.2.14 ibm_deltakernel(double rad, double dilation)	63
5.14.2.15 ibm_findepsilon(int ib, GridObj &g)	63
5.14.2.16 ibm_findsupport(int ib, int m, GridObj &g)	63
5.14.2.17 ibm_initialise(GridObj &g)	63
5.14.2.18 ibm_interpol(int ib, GridObj &g)	63
5.14.2.19 ibm_jacowire(int ib, GridObj &g)	63
5.14.2.20 ibm_move_bodies(GridObj &g)	63
5.14.2.21 ibm_position_update(int ib, GridObj &g)	63
5.14.2.22 ibm_position_update_grp(int group, GridObj &g)	63
5.14.2.23 ibm_spread(int ib, GridObj &g)	63
5.14.2.24 io_readInCloud(PCpts *_PCpts, eObjectType objtype)	63
5.14.2.25 io_restart(bool IO_flag, int level)	63
5.14.2.26 io_vtk_IBwriter(double tval)	64
5.14.2.27 io_write_body_pos(int timestep)	64

5.14.2.28 <code>io_write_lift_drag(int timestep)</code>	64
5.14.2.29 <code>io_writeForceOnObject(double tval)</code>	64
5.14.3 Friends And Related Function Documentation	64
5.14.3.1 <code>GridObj</code>	64
5.15 <code>PCpts</code> Class Reference	65
5.15.1 Detailed Description	65
5.15.2 Constructor & Destructor Documentation	65
5.15.2.1 <code>PCpts(void)</code>	65
5.15.2.2 <code>~PCpts(void)</code>	65
5.15.3 Member Data Documentation	66
5.15.3.1 <code>x</code>	66
5.15.3.2 <code>y</code>	66
5.15.3.3 <code>z</code>	66
5.16 <code>MpiManager::phdf5_struct</code> Struct Reference	66
5.16.1 Detailed Description	67
5.16.2 Member Data Documentation	67
5.16.2.1 <code>halo_max</code>	67
5.16.2.2 <code>halo_min</code>	67
5.16.2.3 <code>i_end</code>	67
5.16.2.4 <code>i_start</code>	67
5.16.2.5 <code>j_end</code>	67
5.16.2.6 <code>j_start</code>	67
5.16.2.7 <code>k_end</code>	67
5.16.2.8 <code>k_start</code>	67
5.16.2.9 <code>level</code>	68
5.16.2.10 <code>region</code>	68
5.16.2.11 <code>writable_data_count</code>	68

6 File Documentation	69
6.1 BFLBody.cpp File Reference	69
6.2 BFLBody.h File Reference	69
6.3 BFLMarker.cpp File Reference	69
6.4 BFLMarker.h File Reference	70
6.5 Body.h File Reference	70
6.6 definitions.h File Reference	70
6.6.1 Macro Definition Documentation	74
6.6.1.1 L_a_x	74
6.6.1.2 L_a_y	75
6.6.1.3 L_a_z	75
6.6.1.4 L_b_x	75
6.6.1.5 L_b_y	75
6.6.1.6 L_b_z	75
6.6.1.7 L_bfl_length	75
6.6.1.8 L_bfl_length_ref	75
6.6.1.9 L_bfl_on_grid_lev	75
6.6.1.10 L_bfl_on_grid_reg	75
6.6.1.11 L_bfl_scale_direction	75
6.6.1.12 L_block_on_grid_lev	76
6.6.1.13 L_block_on_grid_reg	76
6.6.1.14 L_block_x_max	76
6.6.1.15 L_block_x_min	76
6.6.1.16 L_block_y_max	76
6.6.1.17 L_block_y_min	76
6.6.1.18 L_block_z_max	76
6.6.1.19 L_block_z_min	76
6.6.1.20 L_BUILD_FOR_MPI	76
6.6.1.21 L_centre_bfl_z	76
6.6.1.22 L_centre_ibb_z	77

6.6.1.23	<code>L_centre_object_z</code>	77
6.6.1.24	<code>L_CHEAP_NEAREST_NODE_DETECTION</code>	77
6.6.1.25	<code>L_dims</code>	77
6.6.1.26	<code>L_end_BC</code>	77
6.6.1.27	<code>L_FREESTREAM_TUNNEL</code>	77
6.6.1.28	<code>L_grav_direction</code>	77
6.6.1.29	<code>L_grav_force</code>	77
6.6.1.30	<code>L_HDF5_OUTPUT</code>	77
6.6.1.31	<code>L_ibt_angle_horz</code>	77
6.6.1.32	<code>L_ibt_angle_vert</code>	78
6.6.1.33	<code>L_ibt_d</code>	78
6.6.1.34	<code>L_ibt_deform</code>	78
6.6.1.35	<code>L_ibt_delta_rho</code>	78
6.6.1.36	<code>L_ibt_EI</code>	78
6.6.1.37	<code>L_ibt_filament_length</code>	78
6.6.1.38	<code>L_ibt_filament_start_x</code>	78
6.6.1.39	<code>L_ibt_filament_start_y</code>	78
6.6.1.40	<code>L_ibt_filament_start_z</code>	78
6.6.1.41	<code>L_ibt_flex_rigid</code>	78
6.6.1.42	<code>L_ibt_l</code>	79
6.6.1.43	<code>L_ibt_length</code>	79
6.6.1.44	<code>L_ibt_length_ref</code>	79
6.6.1.45	<code>L_ibt_on_grid_lev</code>	79
6.6.1.46	<code>L_ibt_on_grid_reg</code>	79
6.6.1.47	<code>L_ibt_r</code>	79
6.6.1.48	<code>L_ibt_scale_direction</code>	79
6.6.1.49	<code>L_ibt_w</code>	79
6.6.1.50	<code>L_ibt_x</code>	79
6.6.1.51	<code>L_ibt_y</code>	79
6.6.1.52	<code>L_ibt_z</code>	80

6.6.1.53	L_INLET_ON	80
6.6.1.54	L_K	80
6.6.1.55	L_M	80
6.6.1.56	L_MPI_dir	80
6.6.1.57	L_N	80
6.6.1.58	L_num_markers	80
6.6.1.59	L_NumLev	80
6.6.1.60	L_NumReg	80
6.6.1.61	L_nVels	80
6.6.1.62	L_object_length	81
6.6.1.63	L_object_length_ref	81
6.6.1.64	L_object_on_grid_lev	81
6.6.1.65	L_object_on_grid_reg	81
6.6.1.66	L_object_scale_direction	81
6.6.1.67	L_out_every	81
6.6.1.68	L_out_every_forces	81
6.6.1.69	L_out_every_probe	81
6.6.1.70	L_OUTLET_ON	81
6.6.1.71	L_output_precision	81
6.6.1.72	L_PI	82
6.6.1.73	L_Re	82
6.6.1.74	L_restart_out_every	82
6.6.1.75	L_rho_in	82
6.6.1.76	L_SOLID_FROM_FILE	82
6.6.1.77	L_start_BC	82
6.6.1.78	L_start_bfl_x	82
6.6.1.79	L_start_bfl_y	82
6.6.1.80	L_start_ibt_x	82
6.6.1.81	L_start_ibt_y	82
6.6.1.82	L_start_object_x	83

6.6.1.83	L_start_object_y	83
6.6.1.84	L_Timesteps	83
6.6.1.85	L_u_0x	83
6.6.1.86	L_u_0y	83
6.6.1.87	L_u_0z	83
6.6.1.88	L_u_max	83
6.6.1.89	L_u_ref	83
6.6.1.90	L_USE_KBC_COLLISION	83
6.6.1.91	L_wall_thickness	83
6.6.1.92	L_Xcores	84
6.6.1.93	L_Ycores	84
6.6.1.94	L_Zcores	84
6.6.1.95	LUMA_VERSION	84
6.6.2	Variable Documentation	84
6.6.2.1	nProbes	84
6.6.2.2	RefXend	84
6.6.2.3	RefXstart	84
6.6.2.4	RefYend	84
6.6.2.5	RefYstart	84
6.6.2.6	RefZend	84
6.6.2.7	RefZstart	84
6.6.2.8	xProbeLims	84
6.6.2.9	yProbeLims	84
6.6.2.10	zProbeLims	85
6.7	GridObj.cpp File Reference	85
6.8	GridObj.h File Reference	85
6.8.1	Enumeration Type Documentation	85
6.8.1.1	eBCType	85
6.8.1.2	eType	86
6.9	GridObj_init_grids.cpp File Reference	86

6.10 GridObj_ops_boundary.cpp File Reference	86
6.11 GridObj_ops_io.cpp File Reference	86
6.12 GridObj_ops_lbm.cpp File Reference	87
6.13 GridUtils.cpp File Reference	87
6.14 GridUtils.h File Reference	87
6.14.1 Enumeration Type Documentation	87
6.14.1.1 eCartesianDirection	87
6.14.1.2 eMinMax	88
6.15 hdf5luma.h File Reference	88
6.15.1 Macro Definition Documentation	89
6.15.1.1 H5_BUILT_AS_DYNAMIC_LIB	89
6.15.1.2 HDF5_EXT_SZIP	89
6.15.1.3 HDF5_EXT_ZLIB	89
6.15.2 Enumeration Type Documentation	89
6.15.2.1 eHdf5SlabType	89
6.15.3 Function Documentation	89
6.15.3.1 hdf5_writeDataSet(hid_t &memspace, hid_t &filespace, hid_t &dataset_id, e↔ Hdf5SlabType slab_type, int N_lim, int M_lim, int K_lim, int N_mod, int M_mod, int K_mod, GridObj *g, T *data, hid_t hdf_datatype, int TL_thickness, MpiManager↔ ::phdf5_struct hdf_data)	89
6.16 IBody.cpp File Reference	90
6.17 IBody.h File Reference	90
6.18 IBMarker.cpp File Reference	90
6.19 IBMarker.h File Reference	90
6.20 IVector.h File Reference	90
6.21 main_lbm.cpp File Reference	91
6.21.1 Function Documentation	91
6.21.1.1 main(int argc, char *argv[])	91
6.22 Marker.h File Reference	91
6.23 Mpi_buffer_pack.cpp File Reference	91
6.24 Mpi_buffer_size_recv.cpp File Reference	92
6.25 Mpi_buffer_size_send.cpp File Reference	92

6.26	Mpi_buffer_unpk.cpp File Reference	92
6.27	MpiManager.cpp File Reference	92
6.28	MpiManager.h File Reference	92
6.28.1	Macro Definition Documentation	93
6.28.1.1	range_i_left	93
6.28.1.2	range_i_right	93
6.28.1.3	range_j_down	93
6.28.1.4	range_j_up	93
6.28.1.5	range_k_back	94
6.28.1.6	range_k_front	94
6.29	ObjectManager.cpp File Reference	94
6.30	ObjectManager.h File Reference	94
6.30.1	Enumeration Type Documentation	94
6.30.1.1	eObjectType	94
6.31	ObjectManager_init_bflbody.cpp File Reference	95
6.32	ObjectManager_init_ibmbody.cpp File Reference	95
6.33	ObjectManager_ops_ibm.cpp File Reference	95
6.34	ObjectManager_ops_ibmflex.cpp File Reference	95
6.34.1	Macro Definition Documentation	95
6.34.1.1	SWAP	95
6.34.1.2	SWAP	95
6.34.1.3	TINY	95
6.35	ObjectManager_ops_io.cpp File Reference	95
6.36	PCpts.h File Reference	96
6.37	stdafx.cpp File Reference	96
6.37.1	Variable Documentation	96
6.37.1.1	c	96
6.37.1.2	cs	96
6.37.1.3	w	97
6.38	stdafx.h File Reference	97
6.38.1	Macro Definition Documentation	97
6.38.1.1	L_IS_NAN	97
6.38.1.2	LUMA_FAILED	98
6.38.2	Variable Documentation	98
6.38.2.1	c	98
6.38.2.2	cs	98
6.38.2.3	w	98

Chapter 1

Main Page

----- Lattice Boltzmann @ The University of Manchester -----

----- L-U-M-A -----

Copyright (C) 2015, 2016 E-mail contact: info@luma.manchester.ac.uk

This software is for academic use only and not available for distribution without written consent.

Chapter 2

Hierarchical Index

2.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Body< MarkerType >	13
Body< BFLMarker >	13
BFLBody	9
Body< IBMarker >	13
IBBody	37
MpiManager::buffer_struct	16
GridObj	17
GridUtils	30
MpiManager::layer_edges	46
Marker	47
BFLMarker	11
IBMarker	41
MarkerData	49
MpiManager	51
ObjectManager	59
PCpts	65
MpiManager::phdf5_struct	66
vector	
IVector< GenTyp >	44
IVector< double >	44
IVector< eType >	44

Chapter 3

Class Index

3.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

BFLBody	BFL body	9
BFLMarker	BFL marker	11
Body< MarkerType >	Generic body class	13
MpiManager::buffer_struct	Structure storing buffers sizes in each direction for particular grid	16
GridObj	Grid class	17
GridUtils	Grid utility class	30
IBBody	Immersed boundary body	37
IBMarker	Immersed boundary marker	41
IVector< GenTyp >	Index-collapsing vector class	44
MpiManager::layer_edges	Structure containing global positions of the edges of halos	46
Marker	Generic marker class	47
MarkerData	Container class to hold marker information	49
MpiManager	MPI Manager class	51
ObjectManager	Object Manager class	59
PCpts	Class to hold point cloud data	65
MpiManager::phdf5_struct	Structure for storing halo information for HDF5	66

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

BFLBody.cpp	69
BFLBody.h	69
BFLMarker.cpp	69
BFLMarker.h	70
Body.h	70
definitions.h	70
GridObj.cpp	85
GridObj.h	85
GridObj_init_grids.cpp	86
GridObj_ops_boundary.cpp	86
GridObj_ops_io.cpp	86
GridObj_ops_lbm.cpp	87
GridUtils.cpp	87
GridUtils.h	87
hdf5luma.h	88
IBBody.cpp	90
IBBody.h	90
IBMarker.cpp	90
IBMarker.h	90
IVector.h	90
main_lbm.cpp	91
Marker.h	91
Mpi_buffer_pack.cpp	91
Mpi_buffer_size_recv.cpp	92
Mpi_buffer_size_send.cpp	92
Mpi_buffer_unpk.cpp	92
MpiManager.cpp	92
MpiManager.h	92
ObjectManager.cpp	94
ObjectManager.h	94
ObjectManager_init_bflbody.cpp	95
ObjectManager_init_ibmbody.cpp	95
ObjectManager_ops_ibm.cpp	95
ObjectManager_ops_ibmflex.cpp	95
ObjectManager_ops_io.cpp	95
PCpts.h	96
stdafx.cpp	96
stdafx.h	97

Chapter 5

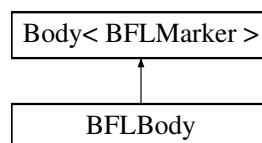
Class Documentation

5.1 BFLBody Class Reference

BFL body.

```
#include <BFLBody.h>
```

Inheritance diagram for BFLBody:



Public Member Functions

- [BFLBody](#) (void)
Default constructor.
- [~BFLBody](#) (void)
Default destructor.
- [BFLBody](#) ([PCpts](#) *_PCpts, [GridObj](#) *g_hierarchy)
Custom constructor to populate body from array of points.

Protected Member Functions

- void [computeQ](#) (int i, int j, int k, [GridObj](#) *g)
Routine to compute wall distance Q.
- void [computeQ](#) (int i, int j, [GridObj](#) *g)
Routine to compute wall distance Q.

Protected Attributes

- `std::vector< std::vector< double > >` [Q](#)
Distance between adjacent lattice site and the surface of the body.

Friends

- class [GridObj](#)

5.1.1 Detailed Description

BFL body.

A BFL body is made up of a collection of BFLMarkers.

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BFLBody::BFLBody (void)

Default constructor.

5.1.2.2 BFLBody::~~BFLBody (void)

Default destructor.

5.1.2.3 BFLBody::BFLBody (PCpts * _PCpts, GridObj * g_hierarchy)

Custom constructor to populate body from array of points.

Parameters

<i>_PCpts</i>	pointer to point cloud data
<i>g_hierarchy</i>	pointer to grid hierarchy

5.1.3 Member Function Documentation

5.1.3.1 void BFLBody::computeQ (int *i*, int *j*, int *k*, GridObj * *g*) [protected]

Routine to compute wall distance Q.

Computes Q values in 3D at a given local voxel for each application of the BFL BC. Performs a line-plane intersection algorithm for every possible triangular plane constructed out of the marker in the voxel and its nearest neighbours.

Parameters

<i>i</i>	local i-index of BFL voxel
<i>j</i>	local j-index of BFL voxel
<i>k</i>	local k-index of BFL voxel
<i>g</i>	pointer to owner grid

5.1.3.2 `void BFLBody::computeQ (int i, int j, GridObj* g)` `[protected]`

Routine to compute wall distance Q.

Computes Q values in 2D at a given local voxel for each application of the BFL BC. Performs a line-line intersection algorithm for each line segment either side of the voxel marker.

Parameters

<i>i</i>	local i-index of BFL voxel
<i>j</i>	local j-index of BFL voxel
<i>g</i>	pointer to owner grid

5.1.4 Friends And Related Function Documentation

5.1.4.1 `friend class GridObj` `[friend]`

5.1.5 Member Data Documentation

5.1.5.1 `std::vector< std::vector<double> > BFLBody::Q` `[protected]`

Distance between adjacent lattice site and the surface of the body.

There are two stores of values. Store 1 is the distance on one side of the wall and store 2 the distance on the other side. One store is appended to the other in this structure.

The documentation for this class was generated from the following files:

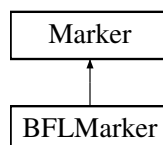
- [BFLBody.h](#)
- [BFLBody.cpp](#)

5.2 BFLMarker Class Reference

BFL marker.

```
#include <BFLMarker.h>
```

Inheritance diagram for BFLMarker:



Public Member Functions

- [BFLMarker](#) (void)
Default constructor.
- [~BFLMarker](#) (void)
Default destructor.
- [BFLMarker](#) (double x, double y, double z)
Custom constructor with position.

Friends

- class [BFLBody](#)

Additional Inherited Members

5.2.1 Detailed Description

BFL marker.

This class declaration is for a BFL Lagrange point. A collection of these points form BFL body.

5.2.2 Constructor & Destructor Documentation

5.2.2.1 [BFLMarker::BFLMarker \(void \)](#)

Default constructor.

5.2.2.2 [BFLMarker::~~BFLMarker \(void \)](#)

Default destructor.

5.2.2.3 [BFLMarker::BFLMarker \(double x, double y, double z \)](#)

Custom constructor with position.

Parameters

<i>x</i>	x-position of marker
<i>y</i>	y-position of marker
<i>z</i>	z-position of marker

5.2.3 Friends And Related Function Documentation

5.2.3.1 friend class BFLBody [friend]

The documentation for this class was generated from the following files:

- [BFLMarker.h](#)
- [BFLMarker.cpp](#)

5.3 Body< MarkerType > Class Template Reference

Generic body class.

```
#include <Body.h>
```

Public Member Functions

- [Body](#) (void)
Default Constructor.
- [~Body](#) (void)
Default destructor.
- [Body](#) (GridObj *g)
Custom constructor setting owning grid.

Protected Member Functions

- void [addMarker](#) (double x, double y, double z)
Add marker to the body.
- [MarkerData](#) * [getMarkerData](#) (double x, double y, double z)
Retrieve marker data.
- void [markerAdder](#) (double x, double y, double z, int &curr_mark, std::vector< int > &counter)
Downsampling marker adding method.
- bool [isInVoxel](#) (double x, double y, double z, int curr_mark)
Determines whether a point is inside another marker's support voxel.
- bool [isVoxelMarkerVoxel](#) (double x, double y, double z)
Determines whether a point is inside an existing marker's support voxel.

Protected Attributes

- double [spacing](#)
Spacing of the markers in physical units.
- std::vector< MarkerType > [markers](#)
Array of markers which make up the body.
- bool [closed_surface](#)
Flag to specify whether or not it is a closed surface (for output)
- [GridObj](#) * [_Owner](#)
Pointer to owning grid.

5.3.1 Detailed Description

```
template<typename MarkerType>
class Body< MarkerType >
```

Generic body class.

Can consist of any type of [Marker](#) so templated.

5.3.2 Constructor & Destructor Documentation

5.3.2.1 `template<typename MarkerType > Body< MarkerType >::Body (void)`

Default Constructor.

5.3.2.2 `template<typename MarkerType > Body< MarkerType >::~~Body (void)`

Default destructor.

5.3.2.3 `template<typename MarkerType > Body< MarkerType >::Body (GridObj * g)`

Custom constructor setting owning grid.

Parameters

<i>g</i>	pointer to grid which owns this body.
----------	---------------------------------------

5.3.3 Member Function Documentation

5.3.3.1 `template<typename MarkerType > void Body< MarkerType >::addMarker (double x, double y, double z)`
[protected]

Add marker to the body.

Parameters

<i>x</i>	X-position of marker.
<i>y</i>	Y-position of marker.
<i>z</i>	Z-position of marker.

5.3.3.2 `template<typename MarkerType > MarkerData * Body< MarkerType >::getMarkerData (double x, double y, double z)` [protected]

Retrieve marker data.

Return marker and voxel/primary support data associated with supplied global position.

Parameters

<i>x</i>	X-position nearest to marker to be retrieved.
<i>y</i>	Y-position nearest to marker to be retrieved.
<i>z</i>	Z-position nearest to marker to be retrieved.

Returns

[MarkerData](#) marker data structure returned. If no marker found, structure is marked as invalid.

5.3.3.3 `template<typename MarkerType > bool Body< MarkerType >::isInVoxel (double x, double y, double z, int curr_mark)` [protected]

Determines whether a point is inside another marker's support voxel.

Parameters

<i>x</i>	X-position of point.
<i>y</i>	Y-position of point.
<i>z</i>	Z-position of point.
<i>curr_mark</i>	ID of the marker.

Returns

true of false

5.3.3.4 `template<typename MarkerType > bool Body< MarkerType >::isVoxelMarkerVoxel (double x, double y, double z)` [protected]

Determines whether a point is inside an existing marker's support voxel.

Parameters

<i>x</i>	X-position of point.
<i>y</i>	Y-position of point.
<i>z</i>	Z-position of point.

Returns

true of false

5.3.3.5 `template<typename MarkerType > void Body< MarkerType >::markerAdder (double x, double y, double z, int & curr_mark, std::vector< int > & counter)` [protected]

Downsampling marker adding method.

This method tries to add a marker to body at the location given but obeys the rules of a voxel-grid filter to ensure markers are distributed such that their spacing roughly matches the background lattice.

Parameters

<i>x</i>	desired X-position of new marker.
<i>y</i>	desired Y-position of new marker.
<i>z</i>	desired Z-position of new marker.
<i>curr_mark</i>	is a reference to the ID of last marker.
<i>counter</i>	is a reference to the total number of markers in the body.

5.3.4 Member Data Documentation

5.3.4.1 `template<typename MarkerType> GridObj* Body< MarkerType >::_Owner` `[protected]`

Pointer to owning grid.

5.3.4.2 `template<typename MarkerType> bool Body< MarkerType >::closed_surface` `[protected]`

Flag to specify whether or not it is a closed surface (for output)

5.3.4.3 `template<typename MarkerType> std::vector<MarkerType> Body< MarkerType >::markers` `[protected]`

Array of markers which make up the body.

5.3.4.4 `template<typename MarkerType> double Body< MarkerType >::spacing` `[protected]`

Spacing of the markers in physical units.

The documentation for this class was generated from the following file:

- [Body.h](#)

5.4 MpiManager::buffer_struct Struct Reference

Structure storing buffers sizes in each direction for particular grid.

```
#include <MpiManager.h>
```

Public Attributes

- int [size](#) [[L_MPI_dir](#)]
Buffer sizes for each direction.
- int [level](#)
Grid level.
- int [region](#)
Region number.

5.4.1 Detailed Description

Structure storing buffers sizes in each direction for particular grid.

5.4.2 Member Data Documentation

5.4.2.1 `int MpiManager::buffer_struct::level`

Grid level.

5.4.2.2 `int MpiManager::buffer_struct::region`

Region number.

5.4.2.3 `int MpiManager::buffer_struct::size[L_MPI_dir]`

Buffer sizes for each direction.

The documentation for this struct was generated from the following file:

- [MpiManager.h](#)

5.5 GridObj Class Reference

Grid class.

```
#include <GridObj.h>
```

Public Member Functions

- [GridObj](#) ()
Default Constructor.
- [GridObj](#) (int level)
Constructor for top level grid.
- [GridObj](#) (int RegionNumber, [GridObj](#) &pGrid)
Constructor for a sub-grid.
- [GridObj](#) (int level, std::vector< int > local_size, std::vector< std::vector< int > > GlobalLimsInd, std::vector< std::vector< double > > GlobalLimsPos)
MPI constructor for top level grid.
- [~GridObj](#) ()
Default Destructor.
- void [LBM_initVelocity](#) ()
Method to initialise the lattice velocity.
- void [LBM_initRho](#) ()
Method to initialise the lattice density.

- void [LBM_initGrid](#) ()
Wrapper to initialise all L0 lattice quantities.
- void [LBM_initGrid](#) (std::vector< int > local_size, std::vector< std::vector< int > > GlobalLimsInd, std::vector< std::vector< double > > GlobalLimsPos)
Method to initialise all L0 lattice quantities.
- void [LBM_initSubGrid](#) (GridObj &pGrid)
Method to initialise all sub-grid quantities.
- void [LBM_initBoundLab](#) ()
Method to initialise wall and object labels on L0.
- void [LBM_initSolidLab](#) ()
Method to initialise label-based solids.
- void [LBM_initRefinedLab](#) (GridObj &pGrid)
Method to initialise all labels on sub-grids.
- void [LBM_init_getInletProfile](#) ()
Method to import an input profile from a file.
- void [LBM_multi](#) (bool IBM_flag)
LBM multi-grid kernel.
- void [LBM_collide](#) ()
Apply collision operator.
- double [LBM_collide](#) (int i, int j, int k, int v)
Equilibrium calculation.
- void [LBM_kbcCollide](#) (int i, int j, int k, IVector< double > &f_new)
KBC collision operator.
- void [LBM_stream](#) ()
Streaming operator.
- void [LBM_macro](#) ()
Macroscopic update.
- void [LBM_macro](#) (int i, int j, int k)
Site-specific macroscopic update.
- void [LBM_boundary](#) (int bc_type_flag)
Method to apply boundary conditions on lattice.
- void [LBM_forcegrid](#) (bool reset_flag)
Method to compute or reset body forces.
- void [bc_applyBounceBack](#) (int label, int i, int j, int k)
Method to apply half-way bounce-back.
- void [bc_applySpecReflect](#) (int label, int i, int j, int k)
Method to apply half-way specular reflection.
- void [bc_applyRegularised](#) (int label, int i, int j, int k)
Method to apply regularised velocity inlet.
- void [bc_applyExtrapolation](#) (int label, int i, int j, int k)
Method to apply extrapolation outlet.
- void [bc_applyBfl](#) (int i, int j, int k)
Method to apply BFL bounce-back.
- void [bc_applyNrbc](#) (int i, int j, int k)
Method to apply NRBC.
- void [bc_solidSiteReset](#) ()
Helper method to set macroscopic quantities of solid sites.
- void [LBM_explode](#) (int RegionNumber)
Explosion operation for pushing information to finer grids.
- void [LBM_coalesce](#) (int RegionNumber)
Coalesce operation for pulling information from finer grids.

- void [LBM_addSubGrid](#) (int RegionNumber)
Wrapper method to add sub-grid to this grid.
- void [io_textout](#) (std::string output_tag)
Verbose ASCII writer.
- void [io_restart](#) (bool IO_flag)
Restart file read-writer.
- void [io_probeOutput](#) ()
Probe writer.
- void [io_lite](#) (double tval, std::string Tag)
ASCII dump of grid data.
- int [io_hdf5](#) (double tval)
HDF5 writer.

Public Attributes

- std::vector< int > [XInd](#)
Vector of global X indices of each site.
- std::vector< int > [YInd](#)
Vector of global Y indices of each site.
- std::vector< int > [ZInd](#)
Vector of global Z indices of each site.
- std::vector< double > [XPos](#)
Vector of global X positions of each site.
- std::vector< double > [YPos](#)
Vector of global Y positions of each site.
- std::vector< double > [ZPos](#)
Vector of global Z positions of each site.
- [IVector](#)< [eType](#) > [LatTyp](#)
Flattened 3D array of site labels.
- int [level](#)
Level in embedded grid hierarchy.
- double [dt](#)
Physical time step size.
- int [t](#)
Number of completed iterations on this level.
- double [nu](#)
Kinematic viscosity (in lattice units)
- double [omega](#)
Relaxation frequency.
- double [timeav_mpi_overhead](#)
Time-averaged time of MPI communication.
- double [timeav_timestep](#)
Time-averaged time of a timestep.
- int [N_lim](#)
Local size of grid in X-direction.
- int [M_lim](#)
Local size of grid in Y-direction.
- int [K_lim](#)
Local size of grid in Z-direction.

Friends

- class [MpiManager](#)
- class [ObjectManager](#)
- class [GridUtils](#)

5.5.1 Detailed Description

Grid class.

This class represents a grid (lattice) and is capable of owning a nested hierarchy of child grids.

5.5.2 Constructor & Destructor Documentation

5.5.2.1 GridObj::GridObj (void)

Default Constructor.

5.5.2.2 GridObj::GridObj (int *level*)

Constructor for top level grid.

Coarse limits are set to zero and then L0-specific initialiser called.

Parameters

<i>level</i>	always should be zero astop level grid.
--------------	---

5.5.2.3 GridObj::GridObj (int *RegionNumber*, GridObj & *pGrid*)

Constructor for a sub-grid.

Parameters

<i>RegionNumber</i>	ID indicating the region of nested refinement to which this sub-grid belongs.
<i>pGrid</i>	pointer to parent grid.

5.5.2.4 GridObj::GridObj (int *level*, std::vector< int > *local_size*, std::vector< std::vector< int > > *GlobalLimsInd*, std::vector< std::vector< double > > *GlobalLimsPos*)

MPI constructor for top level grid.

When using MPI, this constructors a local grid which represents an appropriate portion of the top-level grid as dictated by the extent of this rank.

Parameters

<i>level</i>	always should be zero astop level grid.
<i>local_size</i>	vector indicating dimensions of local grid including halo.
<i>GlobalLimsInd</i>	vector indicating the global indices of the edges of this local grid.
<i>GlobalLimsPos</i>	vector indicating the global positions of the edges of this local grid.

5.5.2.5 GridObj::~GridObj (void)

Default Destructor.

5.5.3 Member Function Documentation

5.5.3.1 void GridObj::bc_applyBfl (int *i*, int *j*, int *k*)

Method to apply BFL bounce-back.

Currently, assumes only 1 BFL body present on the grid.

Parameters

<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.2 void GridObj::bc_applyBounceBack (int *label*, int *i*, int *j*, int *k*)

Method to apply half-way bounce-back.

Parameters

<i>label</i>	current site label.
<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.3 void GridObj::bc_applyExtrapolation (int *label*, int *i*, int *j*, int *k*)

Method to apply extrapolation outlet.

Can only be applied on right-hand wall.

Parameters

<i>label</i>	current site label.
--------------	---------------------

Parameters

<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.4 void GridObj::bc_applyNrbc (int *i*, int *j*, int *k*)

Method to apply NRBC.

Not implemented in this version.

Parameters

<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.5 void GridObj::bc_applyRegularised (int *label*, int *i*, int *j*, int *k*)

Method to apply regularised velocity inlet.

Can be applied on any wall.

Parameters

<i>label</i>	current site label.
<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.6 void GridObj::bc_applySpecReflect (int *label*, int *i*, int *j*, int *k*)

Method to apply half-way specular reflection.

Symmetry boundary condition for free-slip walls.

Parameters

<i>label</i>	current site label.
<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.5.3.7 void GridObj::bc_solidSiteReset ()

Helper method to set macroscopic quantities of solid sites.

Velocity is set to zero and density is set to initial density. Applies to eSolid and eRefinedSolid sites only.

5.5.3.8 int GridObj::io_hdf5 (double *tval*)

HDF5 writer.

Useful grid quantities written out as scalar arrays. One *.h5 file per grid and data is grouped into timesteps within each file.

Parameters

<i>tval</i>	time value being written out.
-------------	-------------------------------

5.5.3.9 void GridObj::io_lite (double *tval*, std::string *TAG*)

ASCII dump of grid data.

Generic ASCII writer for each rank to write out all grid data in rows into a single, unsorted file.

Parameters

<i>tval</i>	time value being written out.
<i>TAG</i>	text identifier for the data.

5.5.3.10 void GridObj::io_probeOutput ()

Probe writer.

This routine writes the quantities at hte probe locations to a single file.

5.5.3.11 void GridObj::io_restart (bool *IO_flag*)

Restart file read-writer.

This routine writes/reads the current rank's data in the custom restart file format. If the file already exists, data is appended. IB body data are also written out but no other body information at present.

Parameters

<i>IO_flag</i>	flag to indicate whether a write (true) or read (false) is required.
----------------	--

5.5.3.12 void GridObj::io_textout (std::string *output_tag*)

Verbose ASCII writer.

Writes all the contents of the grid class at time *t* and call recursively for any sub-grids. Writes to text file "Grids.out" by default.

Parameters

<i>output_tag</i>	text string added to top of output for identification.
-------------------	--

5.5.3.13 void GridObj::LBM_addSubGrid (int *RegionNumber*)

Wrapper method to add sub-grid to this grid.

Parameters

<i>RegionNumber</i>	ID indicating the region of nested refinement to which this sub-grid belongs.
---------------------	---

5.5.3.14 void GridObj::LBM_boundary (int *bc_type_flag*)

Method to apply boundary conditions on lattice.

This method will examine the entire lattice for sites which require a boundary condition but only apply the boundary condition requested in the *bc_type_flag* argument.

Parameters

<i>bc_type_flag</i>	Flag indicating which set of BCs to apply.
---------------------	--

5.5.3.15 void GridObj::LBM_coalesce (int *RegionNumber*)

Coalesce operation for pulling information from finer grids.

Uses the algorithm of Rohde et al. 2006 to pull information from a fine grid TL to a coarse grid TL.

Parameters

<i>RegionNumber</i>	region number of the sub-grid.
---------------------	--------------------------------

5.5.3.16 void GridObj::LBM_collide ()

Apply collision operator.

5.5.3.17 double GridObj::LBM_collide (int *i*, int *j*, int *k*, int *v*)

Equilibrium calculation.

Computes the equilibrium distribution in direction supplied at the given lattice site and returns the value.

Parameters

<i>i</i>	i-index of lattice site.
<i>j</i>	j-index of lattice site.
<i>k</i>	k-index of lattice site.
<i>v</i>	lattice direction.

Returns

equilibrium function.

5.5.3.18 void GridObj::LBM_explode (int *RegionNumber*)

Explosion operation for pushing information to finer grids.

Uses the algorithm of Rohde et al. 2006 to pass information from a coarse grid TL to a fine grid TL.

Parameters

<i>RegionNumber</i>	region number of the sub-grid.
---------------------	--------------------------------

5.5.3.19 void GridObj::LBM_forcegrid (bool *reset_flag*)

Method to compute or reset body forces.

Takes Cartesian force vector and populates forces for each lattice direction. If *reset_flag* is true, then resets the force vectors to zero.

Parameters

<i>reset_flag</i>	flag to indicate whether force vectors should simply be reset.
-------------------	--

5.5.3.20 void GridObj::LBM_init_getInletProfile ()

Method to import an input profile from a file.

Input data may be over- or under-sampled but it must span the physical dimensions of the inlet otherwise the software does not know how to scale the data to fit. Inlet profile is always assumed to be oriented vertically (y-direction).

5.5.3.21 void GridObj::LBM_initBoundLab ()

Method to initialise wall and object labels on L0.

The virtual wind tunnel definitions are implemented by this method.

5.5.3.22 void GridObj::LBM_initGrid ()

Wrapper to initialise all L0 lattice quantities.

This method wraps the MPI-specific version. It is called by the serial build and sets the MPI-specific arguments to default values before calling the full initialiser.

5.5.3.23 void GridObj::LBM_initGrid (std::vector< int > *local_size*, std::vector< std::vector< int > > *global_edge_ind*, std::vector< std::vector< double > > *global_edge_pos*)

Method to initialise all L0 lattice quantities.

Parameters

<i>local_size</i>	local grid size on this rank including halo.
<i>global_edge_ind</i>	global indices of the rank edges.
<i>global_edge_pos</i>	global positions of the rank edges.

5.5.3.24 void GridObj::LBM_initRefinedLab (GridObj & *pGrid*)

Method to initialise all labels on sub-grids.

Boundary labels are set by considering parent labels on overlapping sites and then assigning child labels appropriately.

Parameters

<i>pGrid</i>	reference to parent grid.
--------------	---------------------------

5.5.3.25 void GridObj::LBM_initRho ()

Method to initialise the lattice density.

5.5.3.26 void GridObj::LBM_initSolidLab ()

Method to initialise label-based solids.

5.5.3.27 void GridObj::LBM_initSubGrid (GridObj & *pGrid*)

Method to initialise all sub-grid quantities.

Parameters

<i>pGrid</i>	reference to parent grid.
--------------	---------------------------

5.5.3.28 void GridObj::LBM_initVelocity ()

Method to initialise the lattice velocity.

Unless the L_NO_FLOW macro is defined, the initial velocity everywhere will be set to the values specified in the definitions file.

5.5.3.29 void GridObj::LBM_kbcCollide (int *i*, int *j*, int *k*, IVector< double > & *f_new*)

KBC collision operator.

Applies KBC collision operator using the KBC-N4 and KBC-D models in 3D and 2D, respectively.

Parameters

<i>i</i>	i-index of lattice site.
<i>j</i>	j-index of lattice site.
<i>k</i>	k-index of lattice site.
<i>f_new</i>	reference to the temporary, post-collision grid.

5.5.3.30 void GridObj::LBM_macro ()

Macroscopic update.

Updates macroscopic quantities over the lattice. Also updates time-averaged quantities.

5.5.3.31 void GridObj::LBM_macro (int *i*, int *j*, int *k*)

Site-specific macroscopic update.

Overload of macroscopic quantity calculation to allow it to be applied to a single site as used by the MPI unpacking routine to update the values for the next collision step. This routine does not update the time-averaged quantities.

Parameters

<i>i</i>	i-index of lattice site.
<i>j</i>	j-index of lattice site.
<i>k</i>	k-index of lattice site.

5.5.3.32 void GridObj::LBM_multi (bool *IBM_flag*)

LBM multi-grid kernel.

The LBM kernel manages the calling of all IBM and LBM methods on a given grid. In addition, this method also manages the recursive calling of the method on sub-grids and manages the framework for grid-grid interaction.

Parameters

<i>IBM_flag</i>	flag to indicate whether this kernel is a predictor (true) or corrector (false) step when using IBM.
-----------------	--

5.5.3.33 void GridObj::LBM_stream ()

Streaming operator.

Currently, periodic BCs are only applied on L0. Considers site typing as well as grid location when determining viable streaming.

5.5.4 Friends And Related Function Documentation

5.5.4.1 friend class GridUtils [friend]

5.5.4.2 friend class MpiManager [friend]

5.5.4.3 friend class ObjectManager [friend]

5.5.5 Member Data Documentation

5.5.5.1 double GridObj::dt

Physical time step size.

5.5.5.2 int GridObj::K_lim

Local size of grid in Z-direction.

5.5.5.3 IVector<eType> GridObj::LatTyp

Flattened 3D array of site labels.

5.5.5.4 int GridObj::level

Level in embedded grid hierarchy.

5.5.5.5 int GridObj::M_lim

Local size of grid in Y-direction.

5.5.5.6 int GridObj::N_lim

Local size of grid in X-direction.

5.5.5.7 double GridObj::nu

Kinematic viscosity (in lattice units)

5.5.5.8 double GridObj::omega

Relaxation frequency.

5.5.5.9 int GridObj::t

Number of completed iterations on this level.

5.5.5.10 double GridObj::timeav_mpi_overhead

Time-averaged time of MPI communication.

5.5.5.11 double GridObj::timeav_timestep

Time-averaged time of a timestep.

5.5.5.12 std::vector<int> GridObj::XInd

Vector of global X indices of each site.

5.5.5.13 std::vector<double> GridObj::XPos

Vector of global X positions of each site.

5.5.5.14 std::vector<int> GridObj::YInd

Vector of global Y indices of each site.

5.5.5.15 `std::vector<double> GridObj::YPos`

Vector of global Y positions of each site.

5.5.5.16 `std::vector<int> GridObj::ZInd`

Vector of global Z indices of each site.

5.5.5.17 `std::vector<double> GridObj::ZPos`

Vector of global Z positions of each site.

The documentation for this class was generated from the following files:

- [GridObj.h](#)
- [GridObj.cpp](#)
- [GridObj_init_grids.cpp](#)
- [GridObj_ops_boundary.cpp](#)
- [GridObj_ops_io.cpp](#)
- [GridObj_ops_lbm.cpp](#)

5.6 GridUtils Class Reference

Grid utility class.

```
#include <GridUtils.h>
```

Static Public Member Functions

- static int [createOutputDirectory](#) (std::string [path_str](#))
- static std::vector< int > [onespace](#) (int min, int max)
Creates a integer-spaced vector.
- static std::vector< double > [linspace](#) (double min, double max, int n)
- static double [vecnorm](#) (double vec[])
- static double [vecnorm](#) (double val1, double val2)
- static double [vecnorm](#) (double val1, double val2, double val3)
- static double [vecnorm](#) (std::vector< double > vec)
- static std::vector< int > [getFineIndices](#) (int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)
- static std::vector< int > [getCoarseIndices](#) (int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)
- static double [indexToPosition](#) (int index, double dx)
- static double [dotprod](#) (std::vector< double > vec1, std::vector< double > vec2)
- static std::vector< double > [subtract](#) (std::vector< double > a, std::vector< double > b)
- static std::vector< double > [add](#) (std::vector< double > a, std::vector< double > b)
- static std::vector< double > [vecmultiply](#) (double scalar, std::vector< double > vec)
- static std::vector< double > [crossprod](#) (std::vector< double > vec1, std::vector< double > vec2)
- static std::vector< double > [matrix_multiply](#) (const std::vector< std::vector< double > > &A, const std::vector< double > &x)

- static int [getOpposite](#) (int direction)
- static void [getGrid](#) ([GridObj](#) *&Grids, int level, int region, [GridObj](#) *&ptr)
- static std::vector< int > [getVoxInd](#) (double x, double y, double z)
Get global voxel indices.
- static int [getVoxInd](#) (double p)
Get global voxel index.
- static bool [isOverlapPeriodic](#) (int i, int j, int k, const [GridObj](#) &pGrid)
- static bool [isOnThisRank](#) (int gi, int gj, int gk, const [GridObj](#) &pGrid)
- static bool [isOnThisRank](#) (int gl, enum [eCartesianDirection](#) xyz, const [GridObj](#) &pGrid)
- static bool [hasThisSubGrid](#) (const [GridObj](#) &pGrid, int RegNum)
- static bool [isOnSenderLayer](#) (double pos_x, double pos_y, double pos_z)
- static bool [isOnRecvLayer](#) (double pos_x, double pos_y, double pos_z)
- static bool [isOnSenderLayer](#) (double site_position, enum [eCartesianDirection](#) xyz, enum [eMinMax](#) minmax)
- static bool [isOnRecvLayer](#) (double site_position, enum [eCartesianDirection](#) xyz, enum [eMinMax](#) minmax)
- static bool [isOffGrid](#) (int i, int j, int k, [GridObj](#) &g)
- template<typename NumType >
static NumType [vecnorm](#) (NumType a1, NumType a2, NumType a3)
Computes the L2-norm.
- template<typename NumType >
static NumType [vecnorm](#) (NumType a1, NumType a2)
Computes the L2-norm.
- template<typename NumType >
static NumType [upToZero](#) (NumType x)
Rounds a negative value up to zero.
- template<typename NumType >
static NumType [downToLimit](#) (NumType x, NumType limit)
Rounds a value greater than a limit down to this value.
- template<typename NumType >
static NumType [factorial](#) (NumType n)
Computes the factorial of the supplied value.
- template<typename NumType >
static void [stridedCopy](#) (NumType *dest, NumType *src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset=0)
Performs a strided memcopy.
- template<typename NumType >
static void [global_to_local](#) (int i, int j, int k, [GridObj](#) *g, std::vector< NumType > &locals)
Maps global indices to local indices.
- template<typename NumType >
static void [local_to_global](#) (int i, int j, int k, [GridObj](#) *g, std::vector< NumType > &globals)
Maps local indices to global indices.

Static Public Attributes

- static std::ofstream * [logfile](#)
Handle to output file.
- static std::string [path_str](#)
Static string representing output path.
- static const int [dir_reflect](#) [[L_dims](#) *2][[L_nVels](#)]
Array with hardcoded direction numbering for specular reflection.

5.6.1 Detailed Description

Grid utility class.

Class provides grid utilities including commonly used logical tests. This is a static class and so there is no need to instantiate it.

5.6.2 Member Function Documentation

5.6.2.1 `std::vector< double > GridUtils::add (std::vector< double > a, std::vector< double > b)` `[static]`

5.6.2.2 `int GridUtils::createOutputDirectory (std::string path_str)` `[static]`

5.6.2.3 `std::vector< double > GridUtils::crossprod (std::vector< double > vec1, std::vector< double > vec2)`
`[static]`

5.6.2.4 `double GridUtils::dotprod (std::vector< double > vec1, std::vector< double > vec2)` `[static]`

5.6.2.5 `template<typename NumType > static NumType GridUtils::downToLimit (NumType x, NumType limit)`
`[inline], [static]`

Rounds a value greater than a limit down to this value.

If value is less than or equal to the limit, return the value unchanged.

Parameters

<i>x</i>	value to be rounded
<i>limit</i>	value to be rounded down to

Returns

NumType rounded value

5.6.2.6 `template<typename NumType > static NumType GridUtils::factorial (NumType n)` `[inline], [static]`

Computes the factorial of the supplied value.

If $n == 0$ then returns 1.

Parameters

<i>n</i>	factorial
----------	-----------

Returns

NumType n factorial

5.6.2.7 `std::vector< int > GridUtils::getCoarseIndices (int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)` `[static]`

5.6.2.8 `std::vector< int > GridUtils::getFineIndices (int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)` `[static]`

5.6.2.9 `void GridUtils::getGrid (GridObj *& Grids, int level, int region, GridObj *& ptr)` `[static]`

5.6.2.10 `int GridUtils::getOpposite (int direction)` `[static]`

5.6.2.11 `std::vector< int > GridUtils::getVoxInd (double x, double y, double z)` `[static]`

Get global voxel indices.

Will return the voxel indices of the nearest voxel on the lattice for a given point in global space. Can also be used to map positions to indices.

Parameters

<i>x</i>	global x-position.
<i>y</i>	global y-position.
<i>z</i>	global z-position.

Returns

vector of indices of the nearest voxel.

5.6.2.12 `int GridUtils::getVoxInd (double p)` `[static]`

Get global voxel index.

Will return the voxel index of the nearest voxel on the lattice for a given point in global space in a given direction.

Parameters

<i>p</i>	global position.
----------	------------------

Returns

corresponding global index.

5.6.2.13 `template<typename NumType > static void GridUtils::global_to_local (int i, int j, int k, GridObj * g, std::vector< NumType > & locals)` `[inline],[static]`

Maps global indices to local indices.

Takes a vector container and populates it with the local indices where the supplied global site can be found on the grid supplied. If global indices are not found on the supplied grid then local index of -1 is returned.

Parameters

	<i>i</i>	global index
	<i>j</i>	global index
	<i>k</i>	global index
	<i>g</i>	grid on which local indices are required
out	<i>locals</i>	vector container for local indices

5.6.2.14 `bool GridUtils::hasThisSubGrid (const GridObj & pGrid, int RegNum) [static]`

5.6.2.15 `double GridUtils::indexToPosition (int index, double dx) [static]`

5.6.2.16 `bool GridUtils::isOffGrid (int i, int j, int k, GridObj & g) [static]`

5.6.2.17 `bool GridUtils::isOnRecvLayer (double pos_x, double pos_y, double pos_z) [static]`

5.6.2.18 `bool GridUtils::isOnRecvLayer (double site_position, enum eCartesianDirection xyz, enum eMinMax minmax) [static]`

5.6.2.19 `bool GridUtils::isOnSenderLayer (double pos_x, double pos_y, double pos_z) [static]`

5.6.2.20 `bool GridUtils::isOnSenderLayer (double site_position, enum eCartesianDirection xyz, enum eMinMax minmax) [static]`

5.6.2.21 `bool GridUtils::isOnThisRank (int gi, int gj, int gk, const GridObj & pGrid) [static]`

5.6.2.22 `bool GridUtils::isOnThisRank (int gl, enum eCartesianDirection xyz, const GridObj & pGrid) [static]`

5.6.2.23 `bool GridUtils::isOverlapPeriodic (int i, int j, int k, const GridObj & pGrid) [static]`

5.6.2.24 `std::vector< double > GridUtils::linspace (double min, double max, int n) [static]`

5.6.2.25 `template<typename NumType > static void GridUtils::local_to_global (int i, int j, int k, GridObj * g, std::vector< NumType > & globals) [inline],[static]`

Maps local indices to global indices.

Takes a vector container and populates it with the global indices of the supplied local site

Parameters

	<i>i</i>	local index
	<i>j</i>	local index
	<i>k</i>	local index
	<i>g</i>	grid on which global indices are required
out	<i>globals</i>	vector container for global indices

5.6.2.26 `std::vector< double > GridUtils::matrix_multiply (const std::vector< std::vector< double > > & A, const std::vector< double > & x)` `[static]`

5.6.2.27 `std::vector< int > GridUtils::onespace (int min, int max)` `[static]`

Creates a integer-spaced vector.

Details...

5.6.2.28 `template<typename NumType > static void GridUtils::stridedCopy (NumType * dest, NumType * src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset = 0)` `[inline]`, `[static]`

Performs a strided memcpy.

Memcpy() is designed to copy blocks of contiguous memory. Strided copy copies a pattern of contiguous blocks.

Parameters

<i>dest</i>	pointer to start of destination memory
<i>src</i>	pointer to start of source memory
<i>block</i>	size of contiguous block
<i>offset</i>	offset from the start of the source array
<i>stride</i>	number of elements between start of first block and start of second
<i>count</i>	number of blocks in pattern
<i>buf_offset</i>	offset from start of destination buffer to start writing. Default is zero if not supplied.

5.6.2.29 `std::vector< double > GridUtils::subtract (std::vector< double > a, std::vector< double > b)` `[static]`

5.6.2.30 `template<typename NumType > static NumType GridUtils::upToZero (NumType x)` `[inline]`, `[static]`

Rounds a negative value up to zero.

If value is positive, return the value unchanged.

Parameters

<i>x</i>	value to be rounded
----------	---------------------

Returns

NumType rounded value

5.6.2.31 `std::vector< double > GridUtils::vecmultiply (double scalar, std::vector< double > vec)` `[static]`

5.6.2.32 `double GridUtils::vecnorm (double vec[])` `[static]`

5.6.2.33 `double GridUtils::vecnorm (double val1, double val2) [static]`

5.6.2.34 `double GridUtils::vecnorm (double val1, double val2, double val3) [static]`

5.6.2.35 `double GridUtils::vecnorm (std::vector< double > vec) [static]`

5.6.2.36 `template<typename NumType > static NumType GridUtils::vecnorm (NumType a1, NumType a2, NumType a3)
[inline], [static]`

Computes the L2-norm.

Parameters

<i>a1</i>	first component of the vector
<i>a2</i>	second component of the vector
<i>a3</i>	third component of the vector

Returns

NumType scalar quantity

5.6.2.37 `template<typename NumType > static NumType GridUtils::vecnorm (NumType a1, NumType a2) [inline],
[static]`

Computes the L2-norm.

Parameters

<i>a1</i>	first component of the vector
<i>a2</i>	second component of the vector

Returns

NumType scalar quantity

5.6.3 Member Data Documentation

5.6.3.1 `const int GridUtils::dir_reflect [static]`

Initial value:

```
=
{
    {1, 0, 2, 3, 4, 5, 9, 8, 7, 6, 10, 11, 12, 13, 16, 17, 14, 15, 18},
    {1, 0, 2, 3, 4, 5, 9, 8, 7, 6, 10, 11, 12, 13, 16, 17, 14, 15, 18},
    {0, 1, 3, 2, 4, 5, 8, 9, 6, 7, 13, 12, 11, 10, 14, 15, 16, 17, 18},
    {0, 1, 3, 2, 4, 5, 8, 9, 6, 7, 13, 12, 11, 10, 14, 15, 16, 17, 18},
    {0, 1, 2, 3, 5, 4, 6, 7, 8, 9, 12, 13, 10, 11, 17, 16, 15, 14, 18},
    {0, 1, 2, 3, 5, 4, 6, 7, 8, 9, 12, 13, 10, 11, 17, 16, 15, 14, 18}
}
```

Array with hardcoded direction numbering for specular reflection.

5.6.3.2 `std::ofstream * GridUtils::logfile` [static]

Handle to output file.

5.6.3.3 `std::string GridUtils::path_str` [static]

Static string representing output path.

The documentation for this class was generated from the following files:

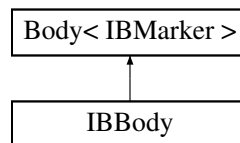
- [GridUtils.h](#)
- [GridObj.cpp](#)
- [GridUtils.cpp](#)
- [main_ibm.cpp](#)

5.7 IBody Class Reference

Immersed boundary body.

```
#include <IBody.h>
```

Inheritance diagram for IBody:



Public Member Functions

- [IBody](#) (void)
Constructor which sets group ID to zero by default.
- [~IBody](#) (void)
Default destructor.
- [IBody](#) (GridObj *g)
Constructor which assigns the owner grid.
- void [addMarker](#) (double x, double y, double z, bool [flex_rigid](#))
Method to add an IB marker to the body.
- void [makeBody](#) (double radius, std::vector< double > centre, bool [flex_rigid](#), bool moving, int group)
Method to seed markers for a sphere / circle.
- void [makeBody](#) (std::vector< double > width_length_depth, std::vector< double > angles, std::vector< double > centre, bool [flex_rigid](#), bool deform, int group)
Method to seed markers for a cuboid / rectangle.
- void [makeBody](#) (int numbermarkers, std::vector< double > start_point, double fil_length, std::vector< double > angles, std::vector< int > [BCs](#), bool [flex_rigid](#), bool deform, int group)
Method to seed markers for a flexible filament.
- double [makeBody](#) (std::vector< double > width_length, double angle, std::vector< double > centre, bool [flex_rigid](#), bool deform, int group, bool plate)
Method to seed markers for a 3D plate inclined from the XZ plane.
- void [makeBody](#) (PCpts *_PCpts)
Method to build a body from a point cloud.

Protected Attributes

- bool [flex_rigid](#)
Flag to indicate flexibility: false == rigid body; true == flexible filament.
- bool [deformable](#)
Flag to indicate deformable body: false == rigid; true == deformable.
- int [groupID](#)
ID of IBbody group – position updates can be driven from a flexible body in a group.
- double [delta_rho](#)
Difference in density between fluid and solid in lattice units.
- double [flexural_rigidity](#)
*Young's modulus E * Second moment of area I .*
- `std::vector< double >` [tension](#)
Tension between the current marker and its neighbour.
- `std::vector< int >` [BCs](#)
BCs type flags (flexible bodies)

Friends

- class [ObjectManager](#)

Additional Inherited Members

5.7.1 Detailed Description

Immersed boundary body.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `IBBody::IBBody (void)`

Constructor which sets group ID to zero by default.

5.7.2.2 `IBBody::~~IBBody (void)`

Default destructor.

5.7.2.3 `IBBody::IBBody (GridObj * g)`

Constructor which assigns the owner grid.

Also sets the group ID to zero.

Parameters

<i>g</i>	pointer to owner grid
----------	-----------------------

5.7.3 Member Function Documentation

5.7.3.1 void IBody::addMarker (double *x*, double *y*, double *z*, bool *flex_rigid*)

Method to add an IB marker to the body.

Adds marker at the given position with the given moving/non-moving flag.

Parameters

<i>x</i>	x-position of marker.
<i>y</i>	y-position of marker.
<i>z</i>	z-position of marker.
<i>flex_rigid</i>	flag to indicate whether marker is movable or not.

5.7.3.2 void IBody::makeBody (double *radius*, std::vector< double > *centre*, bool *flex_rigid*, bool *deform*, int *group*)

Method to seed markers for a sphere / circle.

Parameters

<i>radius</i>	radius of circle/sphere.
<i>centre</i>	position vector of circle/sphere centre.
<i>flex_rigid</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>deform</i>	flag to indicate whether body is movable and requires relocation each time step.
<i>group</i>	ID indicating which group the body is part of for collective operations.

5.7.3.3 void IBody::makeBody (std::vector< double > *width_length_depth*, std::vector< double > *angles*, std::vector< double > *centre*, bool *flex_rigid*, bool *deform*, int *group*)

Method to seed markers for a cuboid / rectangle.

Parameters

<i>width_length_depth</i>	principal dimensions of cuboid / rectangle.
<i>angles</i>	principal orientation of cuboid / rectangle w.r.t. domain axes.
<i>centre</i>	position vector of cuboid / rectangle centre.
<i>flex_rigid</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>deform</i>	flag to indicate whether body is movable and requires relocation each time step.
<i>group</i>	ID indicating which group the body is part of for collective operations.

5.7.3.4 void IBBody::makeBody (int *nummarkers*, std::vector< double > *start_point*, double *fil_length*, std::vector< double > *angles*, std::vector< int > *BCs*, bool *flex_rigid*, bool *deform*, int *group*)

Method to seed markers for a flexible filament.

Parameters

<i>nummarkers</i>	number of markers to use for filament.
<i>start_point</i>	3D position vector of the start of the filament.
<i>fil_length</i>	length of filament in physical units.
<i>angles</i>	two angles representing filament inclination w.r.t. domain axes (horizontal plane and vertical plane).
<i>BCs</i>	vector containing start and end boundary condition types (see class definition for valid values).
<i>flex_rigid</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>deform</i>	flag to indicate whether body is movable and requires relocation each time step.
<i>group</i>	ID indicating which group the body is part of for collective operations.

5.7.3.5 double IBBody::makeBody (std::vector< double > *width_length*, double *angle*, std::vector< double > *centre*, bool *flex_rigid*, bool *deform*, int *group*, bool *plate*)

Method to seed markers for a 3D plate inclined from the XZ plane.

Parameters

<i>width_length</i>	2D vector of principal dimensions of thin plate.
<i>angle</i>	inclination angle from horizontal.
<i>centre</i>	position vector of the plate centre.
<i>flex_rigid</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>deform</i>	flag to indicate whether body is movable and requires relocation each time step.
<i>group</i>	ID indicating which group the body is part of for collective operations.
<i>plate</i>	arbitrary argument to allow overload otherwise would have the same signature as a filament builder.

5.7.3.6 void IBBody::makeBody (PCpts * *_PCpts*)

Method to build a body from a point cloud.

Flexibility and deformable properties taken from definitions.

Parameters

<i>_PCpts</i>	pointer to pointer cloud data.
---------------	--------------------------------

5.7.4 Friends And Related Function Documentation

5.7.4.1 friend class ObjectManager [friend]

5.7.5 Member Data Documentation

5.7.5.1 `std::vector<int> IBBody::BCs` [protected]

BCs type flags (flexible bodies)

5.7.5.2 `bool IBBody::deformable` [protected]

Flag to indicate deformable body: false == rigid; true == deformable.

5.7.5.3 `double IBBody::delta_rho` [protected]

Difference in density between fluid and solid in lattice units.

5.7.5.4 `bool IBBody::flex_rigid` [protected]

Flag to indicate flexibility: false == rigid body; true == flexible filament.

5.7.5.5 `double IBBody::flexural_rigidity` [protected]

Young's modulus E * Second moment of area I .

5.7.5.6 `int IBBody::groupID` [protected]

ID of IBbody group – position updates can be driven from a flexible body in a group.

5.7.5.7 `std::vector<double> IBBody::tension` [protected]

Tension between the current marker and its neighbour.

The documentation for this class was generated from the following files:

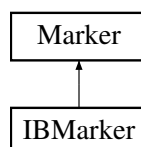
- [IBBody.h](#)
- [IBBody.cpp](#)

5.8 IBMarker Class Reference

Immersed boundary marker.

```
#include <IBMarker.h>
```

Inheritance diagram for IBMarker:



Public Member Functions

- [IBMarker](#) (void)
Default constructor.
- [~IBMarker](#) (void)
Default destructor.
- [IBMarker](#) (double xPos, double yPos, double zPos, bool [flex_rigid](#)=false)
Custom constructor with position.

Protected Attributes

- `std::vector< double >` [fluid_vel](#)
Fluid velocity interpolated from lattice nodes.
- `std::vector< double >` [desired_vel](#)
Desired velocity at marker.
- `std::vector< double >` [force_xyz](#)
Restorative force vector on marker.
- `std::vector< double >` [position_old](#)
Vector containing the physical coordinates (x,y,z) of the marker at t-1. Used for moving bodies.
- `std::vector< double >` [deltaval](#)
Value of delta function for a given support node.
- bool [flex_rigid](#)
Indication as to whether marker is part of a moving or flexible body: false == rigid/fixed; true == flexible/moving.
- double [epsilon](#)
Scaling parameter.
- double [local_area](#)
Area associated with support node in lattice units (same for all points if from same grid and regularly spaced like LBM)
- double [dilation](#)
Dilation parameter in lattice units (same in all directions for uniform Eulerian grid)

Friends

- class [ObjectManager](#)
- class [IBBody](#)

Additional Inherited Members

5.8.1 Detailed Description

Immersed boundary marker.

This class declaration is for an immersed boundary Lagrange point. A collection of these points form an immersed boundary body.

5.8.2 Constructor & Destructor Documentation

5.8.2.1 [IBMarker::IBMarker \(void \)](#) `[inline]`

Default constructor.

5.8.2.2 IBMarker::~IBMarker (void) [inline]

Default destructor.

5.8.2.3 IBMarker::IBMarker (double *xPos*, double *yPos*, double *zPos*, bool *flex_rigid* = false)

Custom constructor with position.

Parameters

<i>xPos</i>	x-position of marker.
<i>yPos</i>	y-position of marker.
<i>zPos</i>	z-position of marker.
<i>flex_rigid</i>	flag to indicate whether marker is movable or not.

5.8.3 Friends And Related Function Documentation

5.8.3.1 friend class IBBody [friend]

5.8.3.2 friend class ObjectManager [friend]

5.8.4 Member Data Documentation

5.8.4.1 std::vector<double> IBMarker::deltaval [protected]

Value of delta function for a given support node.

5.8.4.2 std::vector<double> IBMarker::desired_vel [protected]

Desired velocity at marker.

5.8.4.3 double IBMarker::dilation [protected]

Dilation parameter in lattice units (same in all directions for uniform Eulerian grid)

5.8.4.4 double IBMarker::epsilon [protected]

Scaling parameter.

5.8.4.5 bool IBMarker::flex_rigid [protected]

Indication as to whether marker is part of a moving or flexible body: false == rigid/fixed; true == flexible/moving.

5.8.4.6 `std::vector<double> IBMarker::fluid_vel` [protected]

Fluid velocity interpolated from lattice nodes.

5.8.4.7 `std::vector<double> IBMarker::force_xyz` [protected]

Restorative force vector on marker.

5.8.4.8 `double IBMarker::local_area` [protected]

Area associated with support node in lattice units (same for all points if from same grid and regularly spaced like LBM)

5.8.4.9 `std::vector<double> IBMarker::position_old` [protected]

Vector containing the physical coordinates (x,y,z) of the marker at t-1. Used for moving bodies.

The documentation for this class was generated from the following files:

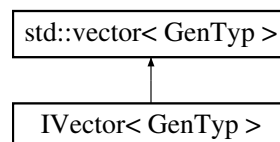
- [IBMarker.h](#)
- [IBMarker.cpp](#)

5.9 `IVector< GenTyp >` Class Template Reference

Index-collapsing vector class.

```
#include <IVector.h>
```

Inheritance diagram for `IVector< GenTyp >`:



Public Member Functions

- [IVector](#) ()
Default constructor.
- [~IVector](#) ()
Default destructor.
- [IVector](#) (size_t size, GenTyp val)
Custom constructor taking type and value.
- GenTyp & [operator\(\)](#) (size_t i, size_t j, size_t k, size_t v, size_t j_max, size_t k_max, size_t v_max)
4D array index flatten.
- GenTyp & [operator\(\)](#) (size_t i, size_t j, size_t k, size_t j_max, size_t k_max)
3D array index flatten.
- GenTyp & [operator\(\)](#) (size_t i, size_t j, size_t j_max)
2D array index flatten.

5.9.1 Detailed Description

```
template<typename GenTyp>
class IVector< GenTyp >
```

Index-collapsing vector class.

This class has all the behaviour of `std::vector` but has a overridden `operator()` to allow automatic flattening of indices before returning a reference of value at indexed location. Needs to be able to accept different datatypes so templated.

5.9.2 Constructor & Destructor Documentation

5.9.2.1 `template<typename GenTyp> IVector< GenTyp >::IVector () [inline]`

Default constructor.

5.9.2.2 `template<typename GenTyp> IVector< GenTyp >::~~IVector () [inline]`

Default destructor.

5.9.2.3 `template<typename GenTyp> IVector< GenTyp >::IVector (size_t size, GenTyp val) [inline]`

Custom constructor taking type and value.

Parameters

<i>size</i>	the desired size of vector
<i>val</i>	the value to fill the new vector with

5.9.3 Member Function Documentation

5.9.3.1 `template<typename GenTyp> GenTyp& IVector< GenTyp >::operator()(size_t i, size_t j, size_t k, size_t v, size_t j_max, size_t k_max, size_t v_max) [inline]`

4D array index flatten.

Override of parentheses to auto-flatten indices to a single index.

Parameters

<i>i</i>	the i index
<i>j</i>	the j index
<i>k</i>	the k index
<i>v</i>	the index in the fourth dimension
<i>j_max</i>	the number of j elements
<i>k_max</i>	the number of k elements
<i>v_max</i>	the number of elements in the fourth dimension

Returns

GenTyp& a reference to the value at this position in the vector

5.9.3.2 `template<typename GenTyp> GenTyp& IVector< GenTyp >::operator() (size_t i, size_t j, size_t k, size_t j_max, size_t k_max) [inline]`

3D array index flatten.

Override of parentheses to auto-flatten indices to a single index.

Parameters

<i>i</i>	the i index
<i>j</i>	the j index
<i>k</i>	the k index
<i>j_max</i>	the number of j elements
<i>k_max</i>	the number of k elements

Returns

GenTyp& a reference to the value at this position in the vector

5.9.3.3 `template<typename GenTyp> GenTyp& IVector< GenTyp >::operator() (size_t i, size_t j, size_t j_max) [inline]`

2D array index flatten.

Parameters

<i>i</i>	the i index
<i>j</i>	the j index
<i>j_max</i>	the number of j elements

Returns

GenTyp& a reference to the value at this position in the vector

The documentation for this class was generated from the following file:

- [IVector.h](#)

5.10 MpiManager::layer_edges Struct Reference

Structure containing global positions of the edges of halos.

```
#include <MpiManager.h>
```

Public Attributes

- double [X](#) [4]
X limits.
- double [Y](#) [4]
Y limits.
- double [Z](#) [4]
Z limits.

5.10.1 Detailed Description

Structure containing global positions of the edges of halos.

Sender (inner) and receiver (outer) parts of halo are located using the convention [left_min left_max right_min right_max] for X,Y and Z.

5.10.2 Member Data Documentation

5.10.2.1 double `MpiManager::layer_edges::X[4]`

X limits.

5.10.2.2 double `MpiManager::layer_edges::Y[4]`

Y limits.

5.10.2.3 double `MpiManager::layer_edges::Z[4]`

Z limits.

The documentation for this struct was generated from the following file:

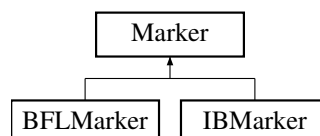
- [MpiManager.h](#)

5.11 Marker Class Reference

Generic marker class.

```
#include <Marker.h>
```

Inheritance diagram for Marker:



Public Member Functions

- [Marker](#) (void)
Default constructor.
- [~Marker](#) (void)
Default destructor.
- [Marker](#) (double x, double y, double z)
Custom constructor which locates marker.

Public Attributes

- `std::vector< double >` [position](#)
Position vector of marker location in physical units.
- `std::vector< int >` [supp_i](#)
X-indices of lattice sites in support of this marker.
- `std::vector< int >` [supp_j](#)
Y-indices of lattice sites in support of this marker.
- `std::vector< int >` [supp_k](#)
Z-indices of lattice sites in support of this marker.
- `std::vector< int >` [support_rank](#)
Array of indices indicating on which rank the given support point resides.

5.11.1 Detailed Description

Generic marker class.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 `Marker::Marker (void)` `[inline]`

Default constructor.

5.11.2.2 `Marker::~~Marker (void)` `[inline]`

Default destructor.

5.11.2.3 `Marker::Marker (double x, double y, double z)` `[inline]`

Custom constructor which locates marker.

Parameters

<code>x</code>	X-position of marker in physical units
<code>y</code>	Y-position of marker in physical units
<code>z</code>	Z-position of marker in physical units

5.11.3 Member Data Documentation

5.11.3.1 `std::vector<double> Marker::position`

Position vector of marker location in physical units.

5.11.3.2 `std::vector<int> Marker::supp_i`

X-indices of lattice sites in support of this marker.

5.11.3.3 `std::vector<int> Marker::supp_j`

Y-indices of lattice sites in support of this marker.

5.11.3.4 `std::vector<int> Marker::supp_k`

Z-indices of lattice sites in support of this marker.

5.11.3.5 `std::vector<int> Marker::support_rank`

Array of indices indicating on which rank the given support point resides.

The documentation for this class was generated from the following file:

- [Marker.h](#)

5.12 MarkerData Class Reference

Container class to hold marker information.

```
#include <Body.h>
```

Public Member Functions

- [MarkerData](#) (int *i*, int *j*, int *k*, double *x*, double *y*, double *z*, int *ID*)
Constructor.
- [MarkerData](#) (void)
Default Constructor.
- [~MarkerData](#) (void)
Default destructor.

Public Attributes

- int `i`
i-index of primary support site
- int `j`
j-index of primary support site
- int `k`
k-index of primary support site
- int `ID`
Marker ID (position in array of markers)
- double `x`
x-position of marker
- double `y`
y-position of marker
- double `z`
z-position of marker

5.12.1 Detailed Description

Container class to hold marker information.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 `MarkerData::MarkerData (int i, int j, int k, double x, double y, double z, int ID)` `[inline]`

Constructor.

Parameters

<i>i</i>	i-index of primary support site
<i>j</i>	j-index of primary support site
<i>k</i>	k-index of primary support site
<i>x</i>	x-position of marker
<i>y</i>	y-position of marker
<i>z</i>	z-position of marker
<i>ID</i>	marker number in a given body

5.12.2.2 `MarkerData::MarkerData (void)` `[inline]`

Default Constructor.

Initialise with invalid marker indicator which is to set the x position to NaN.

5.12.2.3 `MarkerData::~~MarkerData (void)` `[inline]`

Default destructor.

5.12.3 Member Data Documentation

5.12.3.1 `int MarkerData::i`

i-index of primary support site

5.12.3.2 `int MarkerData::ID`

[Marker](#) ID (position in array of markers)

5.12.3.3 `int MarkerData::j`

j-index of primary support site

5.12.3.4 `int MarkerData::k`

k-index of primary support site

5.12.3.5 `double MarkerData::x`

x-position of marker

5.12.3.6 `double MarkerData::y`

y-position of marker

5.12.3.7 `double MarkerData::z`

z-position of marker

The documentation for this class was generated from the following file:

- [Body.h](#)

5.13 MpiManager Class Reference

MPI Manager class.

```
#include <MpiManager.h>
```

Classes

- struct [buffer_struct](#)
Structure storing buffers sizes in each direction for particular grid.
- struct [layer_edges](#)
Structure containing global positions of the edges of halos.
- struct [phdf5_struct](#)
Structure for storing halo information for HDF5.

Public Member Functions

- void [mpi_init](#) ()
Initialisation routine.
- void [mpi_gridbuild](#) ()
Domain decomposition.
- int [mpi_buildCommunicators](#) ()
Define writable sub-grid communicators.
- void [mpi_buffer_pack](#) (int dir, [GridObj](#) *g)
Method to pack the communication buffer.
- void [mpi_buffer_unpack](#) (int dir, [GridObj](#) *g)
Method to unpack the communication buffer.
- void [mpi_buffer_size](#) ()
Pre-calculation of the buffer sizes.
- void [mpi_buffer_size_send](#) ([GridObj](#) *&g)
Method to pre-compute the size of the sender layer buffer.
- void [mpi_buffer_size_recv](#) ([GridObj](#) *&g)
Method to pre-compute the size of the receiver layer buffer.
- void [mpi_writeout_buf](#) (std::string filename, int dir)
Buffer ASCII writer.
- void [mpi_communicate](#) (int level, int regnum)
Communication routine.
- int [mpi_getOpposite](#) (int direction)
Helper method to find opposite direction in MPI topology.

Static Public Member Functions

- static [MpiManager](#) * [getInstance](#) ()
Instance creator.
- static void [destroyInstance](#) ()
Instance destroyer.

Public Attributes

- MPI_Comm [world_comm](#)
Global MPI communicator.
- int [MPI_dims](#) [[L_dims](#)]
Size of MPI Cartesian topology.
- int [neighbour_rank](#) [[L_MPI_dir](#)]
Neighbour rank number for each direction in Cartesian topology.
- int [neighbour_coords](#) [[L_dims](#)][[L_MPI_dir](#)]
Coordinates in MPI topology of neighbour ranks.
- MPI_Comm [subGrid_comm](#) [[L_NumLev](#) * [L_NumReg](#)]
Communicators for sub-grid / region combinations.
- std::vector< [phdf5_struct](#) > [p_data](#)
Vector of structures containing halo descriptors for block writing (HDF5)
- int [global_dims](#) [3]
Global dimensions of problem coarse lattice.
- std::vector< int > [local_size](#)
Dimensions of coarse lattice represented on this rank (includes inner and outer halos).
- std::vector< std::vector< int > > [global_edge_ind](#)
Global indices of coarse lattice nodes represented on this rank.
- std::vector< std::vector< double > > [global_edge_pos](#)
Global positions of coarse lattice nodes represented on this rank.
- [layer_edges](#) [sender_layer_pos](#)
Structure containing sender layer edge positions.
- [layer_edges](#) [recv_layer_pos](#)
Structure containing receiver layer edge positions.
- std::vector< std::vector< double > > [f_buffer_send](#)
Array of resizable outgoing buffers used for data transfer.
- std::vector< std::vector< double > > [f_buffer_recv](#)
Array of resizable incoming buffers used for data transfer.
- MPI_Status [recv_stat](#)
Status structure for Receive return information.
- MPI_Request [send_requests](#) [[L_MPI_dir](#)]
Array of request structures for handles to posted ISends.
- MPI_Status [send_stat](#) [[L_MPI_dir](#)]
Array of statuses for each Isend.
- std::vector< [buffer_struct](#) > [buffer_send_info](#)
Vectors of buffer_info structures holding sender layer size info.
- std::vector< [buffer_struct](#) > [buffer_recv_info](#)
Vectors of buffer_info structures holding receiver layer size info.

Static Public Attributes

- static const int [MPI_cartlab](#) [3][26]
Cartesian unit vectors pointing to each neighbour in Cartesian topology.
- static int [my_rank](#)
Rank number.
- static int [num_ranks](#)
Total number of ranks in MPI Cartesian topology.
- static int [MPI_coords](#) [[L_dims](#)]

Coordinates in MPI Cartesian topology.

- static `GridObj * Grids`

Pointer to grid hierarchy.

- static `std::ofstream * logout`

Logfile handle.

5.13.1 Detailed Description

MPI Manager class.

Class to manage all MPI aspects of the code.

5.13.2 Member Function Documentation

5.13.2.1 `void MpiManager::destroyInstance () [static]`

Instance destroyer.

5.13.2.2 `MpiManager * MpiManager::getInstance () [static]`

Instance creator.

5.13.2.3 `void MpiManager::mpi_buffer_pack (int dir, GridObj * g)`

Method to pack the communication buffer.

Communication buffer is packed with distribution values from the supplied grid. Amount of information is dictated by the direction of the communication being prepared.

Parameters

<i>dir</i>	communication direction.
<i>g</i>	grid doing the communication.

5.13.2.4 `void MpiManager::mpi_buffer_size ()`

Pre-calculation of the buffer sizes.

Wrapper method for computing the buffer sizes for every grid on the rank, both sender and receiver. Must be called post-initialisation.

5.13.2.5 `void MpiManager::mpi_buffer_size_recv (GridObj *& g)`

Method to pre-compute the size of the receiver layer buffer.

A halo consists of a receiver (outer) and sender (inner) layer. This method computes the size of the receiver layers in each communication direction (MPI directions).

Parameters

<i>g</i>	grid being inspected.
----------	-----------------------

5.13.2.6 void MpiManager::mpi_buffer_size_send (GridObj *& *g*)

Method to pre-compute the size of the sender layer buffer.

A halo consists of a receiver (outer) and sender (inner) layer. This method computes the size of the sender layers in each communication direction (MPI directions).

Parameters

<i>g</i>	grid being inspected.
----------	-----------------------

5.13.2.7 void MpiManager::mpi_buffer_unpack (int *dir*, GridObj * *g*)

Method to unpack the communication buffer.

Communication buffer is unpacked onto the supplied grid. Amount and region of unpacking is dictated by the direction of the communication taking place.

Parameters

<i>dir</i>	communication direction.
<i>g</i>	grid doing the communication.

5.13.2.8 int MpiManager::mpi_buildCommunicators ()

Define writable sub-grid communicators.

When using HDF5 in parallel, collective IO operations require all processes to write a non-zero amount of data to the same file. This method examines availability of sub-grid and writable data on the grid (if found) and ensures it is added to a new communicator. Must be called AFTER the grids and buffers have been initialised.

5.13.2.9 void MpiManager::mpi_communicate (int *lev*, int *reg*)

Communication routine.

This method implements the communication between grids of the same level and region across MPI processes. Each call effects communication in all valid directions for the grid of the supplied level and region.

Parameters

<i>lev</i>	level of grid to communicate.
<i>reg</i>	region number of grid to communicate.

5.13.2.10 int MpiManager::mpi_getOpposite (int *direction*)

Helper method to find opposite direction in MPI topology.

The MPI directional vectors do not necessarily correspond to the lattice model direction. The MPI directional vectors are defined separately and hence there is a separate opposite finding method.

Parameters

<i>direction</i>	the outgoing direction whose opposite you wish to find.
------------------	---

5.13.2.11 void MpiManager::mpi_gridbuild ()

Domain decomposition.

Method to decompose the domain and identify local grid sizes. Parameters defined here are used in [GridObj](#) construction.

5.13.2.12 void MpiManager::mpi_init ()

Initialisation routine.

Method is responsible for initialising the MPI topology and associated data. Must be called immediately after MPI_↔init().

5.13.2.13 void MpiManager::mpi_writeout_buf (std::string *filename*, int *dir*)

Buffer ASCII writer.

When verbose MPI logging is turned on this method will write out the communication buffer to an ASCII file.

5.13.3 Member Data Documentation

5.13.3.1 std::vector<buffer_struct> MpiManager::buffer_rcv_info

Vectors of buffer_info structures holding receiver layer size info.

5.13.3.2 std::vector<buffer_struct> MpiManager::buffer_send_info

Vectors of buffer_info structures holding sender layer size info.

5.13.3.3 std::vector< std::vector<double> > MpiManager::f_buffer_rcv

Array of resizeable incoming buffers used for data transfer.

5.13.3.4 `std::vector< std::vector<double> > MpiManager::f_buffer_send`

Array of resizable outgoing buffers used for data transfer.

5.13.3.5 `int MpiManager::global_dims[3]`

Global dimensions of problem coarse lattice.

5.13.3.6 `std::vector< std::vector<int> > MpiManager::global_edge_ind`

Global indices of coarse lattice nodes represented on this rank.

Excludes outer overlapping layer. Rows are x,y,z start and end pairs and columns are rank number.

5.13.3.7 `std::vector< std::vector<double> > MpiManager::global_edge_pos`

Global positions of coarse lattice nodes represented on this rank.

Excluding outer overlapping layer. Rows are x,y,z start and end pairs and columns are rank number.

5.13.3.8 `GridObj * MpiManager::Grids [static]`

Pointer to grid hierarchy.

5.13.3.9 `std::vector<int> MpiManager::local_size`

Dimensions of coarse lattice represented on this rank (includes inner and outer halos).

5.13.3.10 `std::ofstream * MpiManager::logout [static]`

Logfile handle.

5.13.3.11 `const int MpiManager::MPI_cartlab [static]`

Initial value:

```
=
{
    {1, -1, 1, -1, 0, 0, -1, 1, 0, 0, 1, -1, 1, -1, 0, 0, -1, 1, -1, 1, -1, 1, 0,
      0, 1, -1},
    {0, 0, 1, -1, 1, -1, 1, -1, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 0, 0, -1, 1, -1,
      1, -1, 1},
    {0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1, -1, 1,
      -1, 1, -1}
}
```

Cartesian unit vectors pointing to each neighbour in Cartesian topology.

Define 3D such that first 8 mimic the 2D ones. Opposites are simply the next or previous column in the array.

5.13.3.12 `int` `MpiManager::MPI_coords` `[static]`

Coordinates in MPI Cartesian topology.

5.13.3.13 `int` `MpiManager::MPI_dims[L_dims]`

Size of MPI Cartesian topology.

5.13.3.14 `int` `MpiManager::my_rank` `[static]`

Rank number.

5.13.3.15 `int` `MpiManager::neighbour_coords[L_dims][L_MPI_dir]`

Coordinates in MPI topology of neighbour ranks.

5.13.3.16 `int` `MpiManager::neighbour_rank[L_MPI_dir]`

Neighbour rank number for each direction in Cartesian topology.

5.13.3.17 `int` `MpiManager::num_ranks` `[static]`

Total number of ranks in MPI Cartesian topology.

5.13.3.18 `std::vector<phdf5_struct>` `MpiManager::p_data`

Vector of structures containing halo descriptors for block writing (HDF5)

5.13.3.19 `layer_edges` `MpiManager::recv_layer_pos`

Structure containing receiver layer edge positions.

5.13.3.20 `MPI_Status` `MpiManager::recv_stat`

Status structure for Receive return information.

5.13.3.21 `MPI_Request` `MpiManager::send_requests[L_MPI_dir]`

Array of request structures for handles to posted ISends.

5.13.3.22 MPI_Status MpiManager::send_stat[L_MPI_dir]

Array of statuses for each lsend.

5.13.3.23 layer_edges MpiManager::sender_layer_pos

Structure containing sender layer edge positions.

5.13.3.24 MPI_Comm MpiManager::subGrid_comm[L_NumLev * L_NumReg]

Communicators for sub-grid / region combinations.

5.13.3.25 MPI_Comm MpiManager::world_comm

Global MPI communicator.

The documentation for this class was generated from the following files:

- [MpiManager.h](#)
- [GridObj.cpp](#)
- [main_lbm.cpp](#)
- [Mpi_buffer_pack.cpp](#)
- [Mpi_buffer_size_recv.cpp](#)
- [Mpi_buffer_size_send.cpp](#)
- [Mpi_buffer_unpk.cpp](#)
- [MpiManager.cpp](#)

5.14 ObjectManager Class Reference

Object Manager class.

```
#include <ObjectManager.h>
```

Public Member Functions

- void [ibm_apply](#) ([GridObj](#) &g)
- void [ibm_build_body](#) (int body_type)
Builds a prefab immersed boundary body.
- void [ibm_build_body](#) ([PCpts](#) *_PCpts, [GridObj](#) *owner)
Wrapper for building a body from a point cloud.
- void [ibm_initialise](#) ([GridObj](#) &g)
- double [ibm_deltakernel](#) (double rad, double dilation)
- void [ibm_interpol](#) (int ib, [GridObj](#) &g)
- void [ibm_spread](#) (int ib, [GridObj](#) &g)
- void [ibm_findsupport](#) (int ib, int m, [GridObj](#) &g)
- void [ibm_computeforce](#) (int ib, [GridObj](#) &g)
- double [ibm_findepsilon](#) (int ib, [GridObj](#) &g)
- void [ibm_move_bodies](#) ([GridObj](#) &g)
- double [ibm_bicgstab](#) (std::vector< std::vector< double > > &Amatrix, std::vector< double > &bVector, std::vector< double > &epsilon, double tolerance, int maxiterations)
- void [ibm_jacowire](#) (int ib, [GridObj](#) &g)
- void [ibm_position_update](#) (int ib, [GridObj](#) &g)
- void [ibm_position_update_grp](#) (int group, [GridObj](#) &g)
- void [ibm_banbks](#) (double **a, long n, int m1, int m2, double **al, unsigned long indx[], double b[])
- void [ibm_bandec](#) (double **a, long n, int m1, int m2, double **al, unsigned long indx[], double *d)
- void [bfl_build_body](#) (int body_type)
Prefab body building routine.
- void [bfl_build_body](#) ([PCpts](#) *_PCpts)
Wrapper for building BFL body from point cloud.
- void [computeLiftDrag](#) (int i, int j, int k, [GridObj](#) *g)
Compute forces on a rigid object.
- void [io_vtk_IBwriter](#) (double tval)
Write IB body data to VTK file.
- void [io_write_body_pos](#) (int timestep)
Write out position of immersed boundary bodies.
- void [io_write_lift_drag](#) (int timestep)
Write out forces on the markers of immersed boundary bodies.
- void [io_restart](#) (bool IO_flag, int level)
Read/write IB body information to restart file.
- void [io_readInCloud](#) ([PCpts](#) *_PCpts, [eObjectType](#) objtype)
Read in point cloud data.
- void [io_writeForceOnObject](#) (double tval)
Write out the forces on a solid object.

Static Public Member Functions

- static [ObjectManager](#) * [getInstance](#) ()
Get instance method.
- static void [destroyInstance](#) ()
Destroy instance method.
- static [ObjectManager](#) * [getInstance](#) ([GridObj](#) *g)
Overloaded get instance passing in pointer to grid hierarchy.

Friends

- class [GridObj](#)

5.14.1 Detailed Description

Object Manager class.

Class to manage all objects in the domain from creation through manipulation to destruction.

5.14.2 Member Function Documentation

5.14.2.1 void ObjectManager::bfl_build_body (int *body_type*)

Prefab body building routine.

Not implemented in this version.

Parameters

<i>body_type</i>	type of prefab body to be built.
------------------	----------------------------------

5.14.2.2 void ObjectManager::bfl_build_body (PCpts * *_PCpts*)

Wrapper for building BFL body from point cloud.

Parameters

<i>_PCpts</i>	pointer to point cloud data.
---------------	------------------------------

5.14.2.3 void ObjectManager::computeLiftDrag (int *i*, int *j*, int *k*, GridObj * *g*)

Compute forces on a rigid object.

Uses momentum exchange to compute forces on rigid bodies. Currently working with bounce-back objects only. There is no bounding box so if we have walls in the domain they will be counted as well. Also only possible to differentiate between bodies. Lumps all bodies together. identify which body this site relates to so we can differentiate.

Parameters

<i>i</i>	local i-index of solid site.
<i>j</i>	local j-index of solid site.
<i>k</i>	local k-index of solid site.
<i>g</i>	pointer to grid on which object resides.

5.14.2.4 void ObjectManager::destroyInstance () [static]

Destroy instance method.

Instance destuctor.

5.14.2.5 ObjectManager * ObjectManager::getInstance () [static]

Get instance method.

Instance creator.

5.14.2.6 ObjectManager * ObjectManager::getInstance (GridObj * g) [static]

Overloaded get instance passing in pointer to grid hierarchy.

Instance creator with grid hierarchy assignment.

Parameters

<i>g</i>	pointer to grid hierarchy.
----------	----------------------------

5.14.2.7 void ObjectManager::ibm_apply (GridObj & g)

5.14.2.8 void ObjectManager::ibm_banbks (double ** a, long n, int m1, int m2, double ** al, unsigned long indx[], double b[])

5.14.2.9 void ObjectManager::ibm_banded (double ** a, long n, int m1, int m2, double ** al, unsigned long indx[], double * d)

5.14.2.10 double ObjectManager::ibm_bicgstab (std::vector< std::vector< double > > & Amatrix, std::vector< double > & bVector, std::vector< double > & epsilon, double tolerance, int maxiterations)

5.14.2.11 void ObjectManager::ibm_build_body (int body_type)

Builds a prefab immersed boundary body.

Parameters

<i>body_type</i>	type of body to be built.
------------------	---------------------------

5.14.2.12 void ObjectManager::ibm_build_body (PCpts * _PCpts, GridObj * owner)

Wrapper for building a body from a point cloud.

Parameters

<i>_PCpts</i>	pointer to point cloud data.
<i>owner</i>	pointer to the grid on which the body is to be placed.

5.14.2.13 void ObjectManager::ibm_computeforce (int *ib*, GridObj & *g*)

5.14.2.14 double ObjectManager::ibm_deltakernel (double *rad*, double *dilation*)

5.14.2.15 double ObjectManager::ibm_findepsilon (int *ib*, GridObj & *g*)

5.14.2.16 void ObjectManager::ibm_findsupport (int *ib*, int *m*, GridObj & *g*)

5.14.2.17 void ObjectManager::ibm_initialise (GridObj & *g*)

5.14.2.18 void ObjectManager::ibm_interpol (int *ib*, GridObj & *g*)

5.14.2.19 void ObjectManager::ibm_jacowire (int *ib*, GridObj & *g*)

5.14.2.20 void ObjectManager::ibm_move_bodies (GridObj & *g*)

5.14.2.21 void ObjectManager::ibm_position_update (int *ib*, GridObj & *g*)

5.14.2.22 void ObjectManager::ibm_position_update_grp (int *group*, GridObj & *g*)

5.14.2.23 void ObjectManager::ibm_spread (int *ib*, GridObj & *g*)

5.14.2.24 void ObjectManager::io_readInCloud (PCpts * *_PCpts*, eObjectType *objtype*)

Read in point cloud data.

Input data must be in tab separated, 3-column format in the input directory.

Parameters

<i>_PCpts</i>	pointer to empty point cloud data container.
<i>objtype</i>	type of object to be read in.

5.14.2.25 void ObjectManager::io_restart (bool *IO_flag*, int *level*)

Read/write IB body information to restart file.

Parameters

<i>IO_flag</i>	flag indicating write (true) or read (false).
<i>level</i>	level of the grid begin written/read

5.14.2.26 void ObjectManager::io_vtk_IBwriter (double *tval*)

Write IB body data to VTK file.

Currently can only write out un-closed bodies like filaments.

Parameters

<i>tval</i>	time value at which the write out is being performed.
-------------	---

5.14.2.27 void ObjectManager::io_write_body_pos (int *timestep*)

Write out position of immersed boundary bodies.

Parameters

<i>timestep</i>	timestep at which the write out is being performed.
-----------------	---

5.14.2.28 void ObjectManager::io_write_lift_drag (int *timestep*)

Write out forces on the markers of immersed boundary bodies.

Parameters

<i>timestep</i>	timestep at which the write out is being performed.
-----------------	---

5.14.2.29 void ObjectManager::io_writeForceOnObject (double *tval*)

Write out the forces on a solid object.

Writes out the forces on solid objects in the domain computed using momentum exchange. Each rank writes its own file. Output is a CSV file.

Parameters

<i>tval</i>	time value at which write out is taking place.
-------------	--

5.14.3 Friends And Related Function Documentation

5.14.3.1 friend class GridObj [friend]

The documentation for this class was generated from the following files:

- [ObjectManager.h](#)

- [ObjectManager.cpp](#)
- [ObjectManager_init_bflbody.cpp](#)
- [ObjectManager_init_ibmbbody.cpp](#)
- [ObjectManager_ops_ibm.cpp](#)
- [ObjectManager_ops_ibmflex.cpp](#)
- [ObjectManager_ops_io.cpp](#)

5.15 PCpts Class Reference

Class to hold point cloud data.

```
#include <PCpts.h>
```

Public Member Functions

- [PCpts](#) (void)
Default constructor.
- [~PCpts](#) (void)
Default destructor.

Public Attributes

- `std::vector< double > x`
Vector of X positions.
- `std::vector< double > y`
Vector of Y positions.
- `std::vector< double > z`
Vector of Z positions.

5.15.1 Detailed Description

Class to hold point cloud data.

A container class for hold the X, Y and Z positions of points in a point cloud.

5.15.2 Constructor & Destructor Documentation

5.15.2.1 `PCpts::PCpts (void)` `[inline]`

Default constructor.

5.15.2.2 `PCpts::~~PCpts (void)` `[inline]`

Default destructor.

5.15.3 Member Data Documentation

5.15.3.1 `std::vector<double> PCpts::x`

Vector of X positions.

5.15.3.2 `std::vector<double> PCpts::y`

Vector of Y positions.

5.15.3.3 `std::vector<double> PCpts::z`

Vector of Z positions.

The documentation for this class was generated from the following file:

- [PCpts.h](#)

5.16 `MpiManager::phdf5_struct` Struct Reference

Structure for storing halo information for HDF5.

```
#include <MpiManager.h>
```

Public Attributes

- int [i_start](#)
Starting i-index for writable region.
- int [i_end](#)
Ending i-index for writable region.
- int [j_start](#)
Starting j-index for writable region.
- int [j_end](#)
Ending j-index for writable region.
- int [k_start](#)
Starting k-index for writable region.
- int [k_end](#)
Ending k-index for writable region.
- int [halo_min](#)
Size of halo on the top end of a 1D block.
- int [halo_max](#)
Size of halo on the bottom end of a 1D block.
- int [level](#)
Grid level to which these data correspond.
- int [region](#)
Region number to which these data correspond.
- unsigned int [writable_data_count](#) = 0
Writable data count.

5.16.1 Detailed Description

Structure for storing halo information for HDF5.

Structure also stores the amount of writable data on the grid.

5.16.2 Member Data Documentation

5.16.2.1 `int MpiManager::phdf5_struct::halo_max`

Size of halo on the bottom end of a 1D block.

5.16.2.2 `int MpiManager::phdf5_struct::halo_min`

Size of halo on the top end of a 1D block.

5.16.2.3 `int MpiManager::phdf5_struct::i_end`

Ending i-index for writable region.

5.16.2.4 `int MpiManager::phdf5_struct::i_start`

Starting i-index for writable region.

5.16.2.5 `int MpiManager::phdf5_struct::j_end`

Ending j-index for writable region.

5.16.2.6 `int MpiManager::phdf5_struct::j_start`

Starting j-index for writable region.

5.16.2.7 `int MpiManager::phdf5_struct::k_end`

Ending k-index for writable region.

5.16.2.8 `int MpiManager::phdf5_struct::k_start`

Starting k-index for writable region.

5.16.2.9 `int MpiManager::phdf5_struct::level`

Grid level to which these data correspond.

5.16.2.10 `int MpiManager::phdf5_struct::region`

Region number to which these data correspond.

5.16.2.11 `unsigned int MpiManager::phdf5_struct::writable_data_count = 0`

Writable data count.

The documentation for this struct was generated from the following file:

- [MpiManager.h](#)

Chapter 6

File Documentation

6.1 BFLBody.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/BFLBody.h"
#include "../inc/MpiManager.h"
#include "../inc/PCpts.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.2 BFLBody.h File Reference

```
#include "stdafx.h"
#include "Body.h"
#include "BFLMarker.h"
```

Classes

- class [BFLBody](#)
BFL body.

6.3 BFLMarker.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/BFLMarker.h"
#include "../inc/GridUtils.h"
```

6.4 BFLMarker.h File Reference

```
#include "stdafx.h"
#include "Marker.h"
```

Classes

- class [BFLMarker](#)

BFL marker.

6.5 Body.h File Reference

```
#include "stdafx.h"
#include "GridUtils.h"
```

Classes

- class [MarkerData](#)
Container class to hold marker information.
- class [Body](#)< [MarkerType](#) >
Generic body class.

6.6 definitions.h File Reference

```
#include <time.h>
#include <iostream>
#include <fstream>
#include <vector>
#include <iomanip>
#include <math.h>
#include <string>
#include <mpi.h>
```

Macros

- #define [LUMA_VERSION](#) "1.2.0-alpha"
LUMA version.
- #define [L_PI](#) 3.14159265358979323846
PI definition.
- #define [L_BUILD_FOR_MPI](#)
Enable MPI features in build.
- #define [L_out_every](#) 100
How many timesteps before whole grid output.

- `#define L_out_every_forces 10`
Specific output frequency of body forces.
- `#define L_output_precision 4`
Precision of output (for text writers)
- `#define L_HDF5_OUTPUT`
HDF5 dump on output.
- `#define L_out_every_probe 250`
Write out frequency of probe output.
- `#define L_grav_force 1e-10`
Expression for the gravity force.
- `#define L_grav_direction eXDirection`
Gravity direction (specify using enumeration)
- `#define L_restart_out_every 10000`
Frequency of write out of restart file.
- `#define L_USE_KBC_COLLISION`
Use KBC collision operator instead of LBGK by default.
- `#define L_Timesteps 100`
Number of time steps to run simulation for.
- `#define L_Xcores 2`
Number of MPI ranks to divide domain into in X direction.
- `#define L_Ycores 3`
- `#define L_Zcores 2`
- `#define L_dims 3`
Number of dimensions to the problem.
- `#define L_N 80`
Number of x lattice sites.
- `#define L_M 30`
Number of y lattice sites.
- `#define L_K 60`
Number of z lattice sites.
- `#define L_a_x 0`
Start of domain-x.
- `#define L_b_x 8`
End of domain-x.
- `#define L_a_y 0`
Start of domain-y.
- `#define L_b_y 3`
End of domain-y.
- `#define L_a_z 0`
Start of domain-z.
- `#define L_b_z 6`
End of domain-z.
- `#define L_u_ref 0.04`
Reference velocity for scaling, can be mean inlet velocity.
- `#define L_u_max 0.06`
Max velocity of inlet profile.
- `#define L_u_0x L_u_ref`
Initial/inlet x-velocity.
- `#define L_u_0y 0`
Initial/inlet y-velocity.
- `#define L_u_0z 0`

- Initial/inlet z-velocity.*

 - #define `L_rho_in` 1
- Initial density.*

 - #define `L_Re` 10000
- Desired Reynolds number.*

 - #define `L_CHEAP_NEAREST_NODE_DETECTION`
- Perform a nearest-neighbour-type nearest node operation for IBM support calculation.*

 - #define `L_ibb_on_grid_lev` 2
- Provide grid level on which object should be added.*

 - #define `L_ibb_on_grid_reg` 0
- Provide grid region on which object should be added.*

 - #define `L_start_ibb_x` 0.3
- Start X of object bounding box.*

 - #define `L_start_ibb_y` 0.2
- Start Y of object bounding box.*

 - #define `L_centre_ibb_z` 0.5
- Centre of object bounding box in Z direction.*

 - #define `L_ibb_length` 0.5
- The object input is scaled based on this dimension.*

 - #define `L_ibb_scale_direction` `eXDirection`
- Scale in this direction (specify as enumeration)*

 - #define `L_ibb_length_ref` 0.5
- Reference length to be used in the definition of Reynolds number.*

 - #define `L_num_markers` 19
- Number of Lagrange points to use when building a prefab body (approximately)*

 - #define `L_ibb_deform` false
- Default deformable property of body to be built (whether it moves or not)*

 - #define `L_ibb_flex_rigid` false
- Whether a structural calculation needs to be performed on the body.*

 - #define `L_ibb_x` 75.0
- X Position of body centre.*

 - #define `L_ibb_y` 75.0
- Y Position of body centre.*

 - #define `L_ibb_z` 0.0
- Z Position of body centre.*

 - #define `L_ibb_w` 10.0
- Width (x) of IB body.*

 - #define `L_ibb_l` 10.0
- Length (y) of IB body.*

 - #define `L_ibb_d` 0.0
- Depth (z) of IB body.*

 - #define `L_ibb_r` 10.0
- Radius of IB body.*

 - #define `L_ibb_filament_length` 0.2
- Length of filament.*

 - #define `L_ibb_filament_start_x` 0.3
- Start X position of the filament.*

 - #define `L_ibb_filament_start_y` 0.0
- Start Y position of the filament.*

 - #define `L_ibb_filament_start_z` 0.0
- Start Z position of the filament.*

- `#define L_abb_angle_vert 90`
Inclination of filament in XY plane.
- `#define L_abb_angle_horz 0`
Inclination of filament in XZ plane.
- `#define L_start_BC 2`
Type of boundary condition at filament start: 0 == free; 1 = simply supported; 2 == clamped.
- `#define L_end_BC 0`
Type of boundary condition at filament end: 0 == free; 1 = simply supported; 2 == clamped.
- `#define L_abb_delta_rho 1.0`
Difference in density (lattice units) between solid and fluid.
- `#define L_abb_EI 2.0`
Flexural rigidity (lattice units) of filament.
- `#define L_FREESTREAM_TUNNEL`
Adds a inlet to all faces.
- `#define L_INLET_ON`
Turn on inlet boundary (assumed left-hand wall - default Do Nothing)
- `#define L_OUTLET_ON`
Turn on outlet boundary (assumed right-hand wall – default First Order Extrap.)
- `#define L_wall_thickness 1`
Thickness of walls in coarsest lattice units.
- `#define L_block_on_grid_lev 0`
Provide grid level on which block should be added.
- `#define L_block_on_grid_reg 0`
Provide grid region on which block should be added.
- `#define L_block_x_min 32`
Index of start of object/wall in x-direction.
- `#define L_block_x_max 64`
Index of end of object/wall in x-direction.
- `#define L_block_y_min 16`
Index of start of object/wall in y-direction.
- `#define L_block_y_max 48`
Index of end of object/wall in y-direction.
- `#define L_block_z_min 16`
Index of start of object/wall in z-direction.
- `#define L_block_z_max 48`
Index of end of object/wall in z-direction.
- `#define L_SOLID_FROM_FILE`
Build solid body from point cloud file.
- `#define L_object_on_grid_lev 4`
Provide grid level on which object should be added.
- `#define L_object_on_grid_reg 0`
Provide grid region on which object should be added.
- `#define L_start_object_x 40`
Index for start of object bounding box in X direction.
- `#define L_start_object_y 16`
Index for start of object bounding box in Y direction.
- `#define L_centre_object_z 121`
Index for centre of object bounding box in Z direction.
- `#define L_object_length 160`
The object input is scaled based on this dimension.
- `#define L_object_scale_direction eXDirection`

- *Scale in this direction (specify as enumeration)*
- `#define L_object_length_ref 160`
Reference length to be used in the definition of Reynolds number.
- `#define L_bfl_on_grid_lev 1`
Provide grid level on which BFL body should be added.
- `#define L_bfl_on_grid_reg 0`
Provide grid region on which BFL body should be added.
- `#define L_start_bfl_x 50`
Index for start of object bounding box in X direction.
- `#define L_start_bfl_y 100`
Index for start of object bounding box in Y direction.
- `#define L_centre_bfl_z 20`
Index for centre of object bounding box in Z direction.
- `#define L_bfl_length 50`
The BFL object input is scaled based on this dimension.
- `#define L_bfl_scale_direction eXDirection`
Scale in this direction (specify as enumeration)
- `#define L_bfl_length_ref 10`
Reference length to be used in the definition of Reynolds number.
- `#define L_NumLev 4`
Levels of refinement (0 = coarse grid only).
- `#define L_NumReg 1`
Number of refined regions (can be arbitrary if L_NumLev = 0)
- `#define L_nVels 27`
Number of lattice velocities.
- `#define L_MPI_dir 26`
Number of MPI directions.

Variables

- `static const int nProbes [3] = {3, 3, 3}`
Number of probes in each direction (x, y, z)
- `static const int xProbeLims [2] = {90, 270}`
Limits of X plane for array of probes.
- `static const int yProbeLims [2] = {15, 45}`
Limits of Y plane for array of probes.
- `static const int zProbeLims [2] = {30, 120}`
Limits of Z plane for array of probes.
- `static const int RefXstart [L_NumLev][L_NumReg] = { {10}, {5}, {10}, {20} }`
- `static const int RefXend [L_NumLev][L_NumReg] = { {50}, {70}, {110}, {160} }`
- `static const int RefYstart [L_NumLev][L_NumReg] = { {0}, {0}, {0}, {0} }`
- `static const int RefYend [L_NumLev][L_NumReg] = { {15}, {25}, {40}, {60} }`
- `static int RefZstart [L_NumLev][L_NumReg] = { {15}, {5}, {10}, {20} }`
- `static int RefZend [L_NumLev][L_NumReg] = { {45}, {55}, {90}, {140} }`

6.6.1 Macro Definition Documentation

6.6.1.1 `#define L_a_x 0`

Start of domain-x.

6.6.1.2 `#define L_a_y 0`

Start of domain-y.

6.6.1.3 `#define L_a_z 0`

Start of domain-z.

6.6.1.4 `#define L_b_x 8`

End of domain-x.

6.6.1.5 `#define L_b_y 3`

End of domain-y.

6.6.1.6 `#define L_b_z 6`

End of domain-z.

6.6.1.7 `#define L_bfl_length 50`

The BFL object input is scaled based on this dimension.

6.6.1.8 `#define L_bfl_length_ref 10`

Reference length to be used in the definition of Reynolds number.

6.6.1.9 `#define L_bfl_on_grid_lev 1`

Provide grid level on which BFL body should be added.

6.6.1.10 `#define L_bfl_on_grid_reg 0`

Provide grid region on which BFL body should be added.

6.6.1.11 `#define L_bfl_scale_direction eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.12 #define L_block_on_grid_lev 0

Provide grid level on which block should be added.

6.6.1.13 #define L_block_on_grid_reg 0

Provide grid region on which block should be added.

6.6.1.14 #define L_block_x_max 64

Index of end of object/wall in x-direction.

6.6.1.15 #define L_block_x_min 32

Index of start of object/wall in x-direction.

6.6.1.16 #define L_block_y_max 48

Index of end of object/wall in y-direction.

6.6.1.17 #define L_block_y_min 16

Index of start of object/wall in y-direction.

6.6.1.18 #define L_block_z_max 48

Index of end of object/wall in z-direction.

6.6.1.19 #define L_block_z_min 16

Index of start of object/wall in z-direction.

6.6.1.20 #define L_BUILD_FOR_MPI

Enable MPI features in build.

6.6.1.21 #define L_centre_bfl_z 20

Index for centre of object bounding box in Z direction.

6.6.1.22 #define L_centre_ibb_z 0.5

Centre of object bounding box in Z direction.

6.6.1.23 #define L_centre_object_z 121

Index for centre of object bounding box in Z direction.

6.6.1.24 #define L_CHEAP_NEAREST_NODE_DETECTION

Perform a nearest-neighbour-type nearest node operation for IBM support calculation.

6.6.1.25 #define L_dims 3

Number of dimensions to the problem.

6.6.1.26 #define L_end_BC 0

Type of boundary condition at filament end: 0 == free; 1 = simply supported; 2 == clamped.

6.6.1.27 #define L_FREESTREAM_TUNNEL

Adds a inlet to all faces.

6.6.1.28 #define L_grav_direction eXDirection

Gravity direction (specify using enumeration)

6.6.1.29 #define L_grav_force 1e-10

Expression for the gravity force.

6.6.1.30 #define L_HDF5_OUTPUT

HDF5 dump on output.

6.6.1.31 #define L_ibb_angle_horz 0

Inclination of filament in XZ plane.

6.6.1.32 `#define L_abb_angle_vert 90`

Inclination of filament in XY plane.

6.6.1.33 `#define L_abb_d 0.0`

Depth (z) of IB body.

6.6.1.34 `#define L_abb_deform false`

Default deformable property of body to be built (whether it moves or not)

6.6.1.35 `#define L_abb_delta_rho 1.0`

Difference in density (lattice units) between solid and fluid.

6.6.1.36 `#define L_abb_EI 2.0`

Flexural rigidity (lattice units) of filament.

6.6.1.37 `#define L_abb_filament_length 0.2`

Length of filament.

6.6.1.38 `#define L_abb_filament_start_x 0.3`

Start X position of the filament.

6.6.1.39 `#define L_abb_filament_start_y 0.0`

Start Y position of the filament.

6.6.1.40 `#define L_abb_filament_start_z 0.0`

Start Z position of the filament.

6.6.1.41 `#define L_abb_flex_rigid false`

Whether a structural calculation needs to be performed on the body.

6.6.1.42 `#define L_obb_l 10.0`

Length (y) of IB body.

6.6.1.43 `#define L_obb_length 0.5`

The object input is scaled based on this dimension.

6.6.1.44 `#define L_obb_length_ref 0.5`

Reference length to be used in the definition of Reynolds number.

6.6.1.45 `#define L_obb_on_grid_lev 2`

Provide grid level on which object should be added.

6.6.1.46 `#define L_obb_on_grid_reg 0`

Provide grid region on which object should be added.

6.6.1.47 `#define L_obb_r 10.0`

Radius of IB body.

6.6.1.48 `#define L_obb_scale_direction eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.49 `#define L_obb_w 10.0`

Width (x) of IB body.

6.6.1.50 `#define L_obb_x 75.0`

X Position of body centre.

6.6.1.51 `#define L_obb_y 75.0`

Y Position of body centre.

6.6.1.52 `#define L_abb_z 0.0`

Z Position of body centre.

6.6.1.53 `#define L_INLET_ON`

Turn on inlet boundary (assumed left-hand wall - default Do Nothing)

6.6.1.54 `#define L_K 60`

Number of z lattice sites.

6.6.1.55 `#define L_M 30`

Number of y lattice sites.

6.6.1.56 `#define L_MPI_dir 26`

Number of MPI directions.

6.6.1.57 `#define L_N 80`

Number of x lattice sites.

6.6.1.58 `#define L_num_markers 19`

Number of Lagrange points to use when building a prefab body (approximately)

6.6.1.59 `#define L_NumLev 4`

Levels of refinement (0 = coarse grid only).

6.6.1.60 `#define L_NumReg 1`

Number of refined regions (can be arbitrary if L_NumLev = 0)

6.6.1.61 `#define L_nVels 27`

Number of lattice velocities.

6.6.1.62 #define L_object_length 160

The object input is scaled based on this dimension.

6.6.1.63 #define L_object_length_ref 160

Reference length to be used in the definition of Reynolds number.

6.6.1.64 #define L_object_on_grid_lev 4

Provide grid level on which object should be added.

6.6.1.65 #define L_object_on_grid_reg 0

Provide grid region on which object should be added.

6.6.1.66 #define L_object_scale_direction eXDirection

Scale in this direction (specify as enumeration)

6.6.1.67 #define L_out_every 100

How many timesteps before whole grid output.

6.6.1.68 #define L_out_every_forces 10

Specific output frequency of body forces.

6.6.1.69 #define L_out_every_probe 250

Write out frequency of probe output.

6.6.1.70 #define L_OUTLET_ON

Turn on outlet boundary (assumed right-hand wall – default First Order Extrap.)

6.6.1.71 #define L_output_precision 4

Precision of output (for text writers)

6.6.1.72 `#define L_PI 3.14159265358979323846`

PI definition.

6.6.1.73 `#define L_Re 10000`

Desired Reynolds number.

6.6.1.74 `#define L_restart_out_every 10000`

Frequency of write out of restart file.

6.6.1.75 `#define L_rho_in 1`

Initial density.

6.6.1.76 `#define L_SOLID_FROM_FILE`

Build solid body from point cloud file.

6.6.1.77 `#define L_start_BC 2`

Type of boundary condition at filament start: 0 == free; 1 = simply supported; 2 == clamped.

6.6.1.78 `#define L_start_bfl_x 50`

Index for start of object bounding box in X direction.

6.6.1.79 `#define L_start_bfl_y 100`

Index for start of object bounding box in Y direction.

6.6.1.80 `#define L_start_ibb_x 0.3`

Start X of object bounding box.

6.6.1.81 `#define L_start_ibb_y 0.2`

Start Y of object bounding box.

6.6.1.82 #define L_start_object_x 40

Index for start of object bounding box in X direction.

6.6.1.83 #define L_start_object_y 16

Index for start of object bounding box in Y direction.

6.6.1.84 #define L_Timesteps 100

Number of time steps to run simulation for.

6.6.1.85 #define L_u_0x L_u_ref

Initial/inlet x-velocity.

6.6.1.86 #define L_u_0y 0

Initial/inlet y-velocity.

6.6.1.87 #define L_u_0z 0

Initial/inlet z-velocity.

6.6.1.88 #define L_u_max 0.06

Max velocity of inlet profile.

6.6.1.89 #define L_u_ref 0.04

Reference velocity for scaling, can be mean inlet velocity.

6.6.1.90 #define L_USE_KBC_COLLISION

Use KBC collision operator instead of LBGK by default.

6.6.1.91 #define L_wall_thickness 1

Thickness of walls in coarsest lattice units.

6.6.1.92 `#define L_Xcores 2`

Number of MPI ranks to divide domain into in X direction.

6.6.1.93 `#define L_Ycores 3`

Number of MPI ranks to divide domain into in Y direction

6.6.1.94 `#define L_Zcores 2`

Number of MPI ranks to divide domain into in Z direction. Set to 1 if doing a 2D problem when using custom MPI sizes

6.6.1.95 `#define LUMA_VERSION "1.2.0-alpha"`

LUMA version.

6.6.2 Variable Documentation

6.6.2.1 `const int nProbes[3] = {3, 3, 3} [static]`

Number of probes in each direction (x, y, z)

6.6.2.2 `const int RefXend[L_NumLev][L_NumReg] = { {50}, {70}, {110}, {160} } [static]`

6.6.2.3 `const int RefXstart[L_NumLev][L_NumReg] = { {10}, {5}, {10}, {20} } [static]`

6.6.2.4 `const int RefYend[L_NumLev][L_NumReg] = { {15}, {25}, {40}, {60} } [static]`

6.6.2.5 `const int RefYstart[L_NumLev][L_NumReg] = { {0}, {0}, {0}, {0} } [static]`

6.6.2.6 `int RefZend[L_NumLev][L_NumReg] = { {45}, {55}, {90}, {140} } [static]`

6.6.2.7 `int RefZstart[L_NumLev][L_NumReg] = { {15}, {5}, {10}, {20} } [static]`

6.6.2.8 `const int xProbeLims[2] = {90, 270} [static]`

Limits of X plane for array of probes.

6.6.2.9 `const int yProbeLims[2] = {15, 45} [static]`

Limits of Y plane for array of probes.

6.6.2.10 `const int zProbeLims[2] = {30, 120} [static]`

Limits of Z plane for array of probes.

6.7 GridObj.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/MpiManager.h"
#include "../inc/GridUtils.h"
```

6.8 GridObj.h File Reference

```
#include "stdafx.h"
#include "IVector.h"
```

Classes

- class [GridObj](#)
Grid class.

Enumerations

- enum [eType](#) {
 [eSolid](#), [eFluid](#), [eRefined](#), [eTransitionToCoarser](#),
 [eTransitionToFiner](#), [eBFL](#), [eSymmetry](#), [eInlet](#),
 [eOutlet](#), [eRefinedSolid](#), [eRefinedSymmetry](#), [eRefinedInlet](#) }
 Lattice typing labels.
- enum [eBCType](#) {
 [eBCAll](#), [eBCSolidSymmetry](#), [eBCInlet](#), [eBCOutlet](#),
 [eBCInletOutlet](#), [eBCBFL](#) }
 Flag for indicating which BCs to apply.

6.8.1 Enumeration Type Documentation

6.8.1.1 enum [eBCType](#)

Flag for indicating which BCs to apply.

Enumerator

- [eBCAll](#)** Apply all BCs.
- [eBCSolidSymmetry](#)** Apply just solid and symmetry BCs.
- [eBCInlet](#)** Apply just inlet BCs.
- [eBCOutlet](#)** Apply just outlet BCs.
- [eBCInletOutlet](#)** Apply inlet and outlet BCs.
- [eBCBFL](#)** Apply just BFL BCs.

6.8.1.2 enum eType

Lattice typing labels.

Enumerator

- eSolid** Rigid, solid site.
- eFluid** Fluid site.
- eRefined** Fluid site which is represented on a finer grid.
- eTransitionToCoarser** Fluid site coupled to a coarser grid.
- eTransitionToFiner** Fluid site coupled to a finer grid.
- eBFL** Site containing a BFL marker.
- eSymmetry** Symmetry boundary.
- eInlet** Inlet boundary.
- eOutlet** Outlet boundary.
- eRefinedSolid** Rigid, solid site represented on a finer grid.
- eRefinedSymmetry** Symmetry boundary represented on a finer grid.
- eRefinedInlet** Inlet site represented on a finer grid.

6.9 GridObj_init_grids.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/MpiManager.h"
#include "../inc/GridUtils.h"
```

6.10 GridObj_ops_boundary.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/BFLBody.h"
#include "../inc/ObjectManager.h"
#include "../inc/GridUtils.h"
```

6.11 GridObj_ops_io.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/MpiManager.h"
#include "../inc/ObjectManager.h"
#include "../inc/GridUtils.h"
#include "../inc/hdf5luma.h"
```

6.12 GridObj_ops_lbm.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/IVector.h"
#include "../inc/ObjectManager.h"
#include "../inc/MpiManager.h"
#include "../inc/GridUtils.h"
```

6.13 GridUtils.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridUtils.h"
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
```

6.14 GridUtils.h File Reference

```
#include "stdafx.h"
#include "GridObj.h"
```

Classes

- class [GridUtils](#)
Grid utility class.

Enumerations

- enum [eCartesianDirection](#) { [eXDirection](#), [eYDirection](#), [eZDirection](#) }
Enumeration for directional options.
- enum [eMinMax](#) { [eMinimum](#), [eMaximum](#) }
Enumeration for minimum and maximum.

6.14.1 Enumeration Type Documentation

6.14.1.1 enum eCartesianDirection

Enumeration for directional options.

Enumerator

eXDirection X-direction.
eYDirection Y-direction.
eZDirection Z-direction.

6.14.1.2 enum eMinMax

Enumeration for minimum and maximum.

Some utility methods need to know whether they should be looking at or for a maximum or minimum edge of a grid so we use this enumeration to specify.

Enumerator

eMinimum Minimum.

eMaximum Maximum.

6.15 hdf5luma.h File Reference

```
#include "stdafx.h"
#include "hdf5.h"
#include "MpiManager.h"
```

Macros

- #define [H5_BUILT_AS_DYNAMIC_LIB](#)
- #define [HDF5_EXT_ZLIB](#)
- #define [HDF5_EXT_SZIP](#)

Enumerations

- enum [eHdf5SlabType](#) {
[eScalar](#), [eVector](#), [eProductVector](#), [ePosX](#),
[ePosY](#), [ePosZ](#) }

Defines the type of storage arrangement of the variable in memory.

Functions

- template<typename T >
void [hdf5_writeDataSet](#) (hid_t &memspace, hid_t &filespace, hid_t &dataset_id, [eHdf5SlabType](#) slab_type,
int N_lim, int M_lim, int K_lim, int N_mod, int M_mod, int K_mod, [GridObj](#) *g, T *data, hid_t hdf_datatype, int
TL_thickness, [MpiManager::phdf5_struct](#) hdf_data)

Helper method to write out using HDF5.

6.15.1 Macro Definition Documentation

6.15.1.1 `#define H5_BUILT_AS_DYNAMIC_LIB`

6.15.1.2 `#define HDF5_EXT_SZIP`

6.15.1.3 `#define HDF5_EXT_ZLIB`

6.15.2 Enumeration Type Documentation

6.15.2.1 `enum eHdf5SlabType`

Defines the type of storage arrangement of the variable in memory.

The write wrapper can then extract the data from memory and write it to an HDF5 file using a particular hyperslab selection.

Enumerator

- eScalar** 2/3D data – One variable per grid site
- eVector** 2/3D data – L_dims variables per grid site
- eProductVector** 1D data – 3*L_dims-3 variables per grid site
- ePosX** 1D data – Single L_dim vector per dimension
- ePosY** 1D data – Single L_dim vector per dimension
- ePosZ** 1D data – Single L_dim vector per dimension

6.15.3 Function Documentation

6.15.3.1 `template<typename T> void hdf5_writeDataSet (hid_t & memspace, hid_t & filespace, hid_t & dataset_id, eHdf5SlabType slab_type, int N_lim, int M_lim, int K_lim, int N_mod, int M_mod, int K_mod, GridObj * g, T * data, hid_t hdf_datatype, int TL_thickness, MpiManager::phdf5_struct hdf_data)`

Helper method to write out using HDF5.

Automatically selects the correct slab arrangement and buffers the data accordingly before writing to structured file.

Parameters

<i>memspace</i>	memory dataspace id
<i>filespace</i>	file dataspace id
<i>dataset_id</i>	dataset id
<i>slab_type</i>	slab type enum
<i>N_lim</i>	number of X-direction sites on the local grid
<i>M_lim</i>	number of Y-direction sites on the local grid
<i>K_lim</i>	number of Z-direction sites on the local grid
<i>N_mod</i>	number of X-direction sites excluding TL sites
<i>M_mod</i>	number of Y-direction sites excluding TL sites
<i>K_mod</i>	number of Z-direction sites excluding TL sites
<i>g</i>	pointer to grid which we are writing out
<i>data</i>	pointer to the start of the array to be written
<i>hdf_datatype</i>	HDF5 datatype being written
<i>TL_thickness</i>	the thickness of the TL on this grid level in local lattice units
<i>hdf_data</i>	the data structure containing information about local halos

6.16 IBody.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/IBody.h"
#include "../inc/IBMarker.h"
#include "../inc/PCpts.h"
#include "../inc/GridUtils.h"
```

6.17 IBody.h File Reference

```
#include "stdafx.h"
#include "Body.h"
```

Classes

- class [IBody](#)
Immersed boundary body.

6.18 IBMarker.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/IBMarker.h"
```

6.19 IBMarker.h File Reference

```
#include "stdafx.h"
#include "Marker.h"
```

Classes

- class [IBMarker](#)
Immersed boundary marker.

6.20 IVector.h File Reference

```
#include "stdafx.h"
```

Classes

- class [IVector< GenTyp >](#)
Index-collapsing vector class.

6.21 main_lbm.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/MpiManager.h"
#include "../inc/ObjectManager.h"
#include "../inc/GridUtils.h"
#include "../inc/PCpts.h"
```

Functions

- int [main](#) (int argc, char *argv[])
Entry point for the application.

6.21.1 Function Documentation

6.21.1.1 int main (int argc, char * argv[])

Entry point for the application.

6.22 Marker.h File Reference

```
#include "stdafx.h"
```

Classes

- class [Marker](#)
Generic marker class.

6.23 Mpi_buffer_pack.cpp File Reference

```
#include "../inc/stdafx.h"
#include <mpi.h>
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.24 `Mpi_buffer_size_recv.cpp` File Reference

```
#include "../inc/stdafx.h"
#include <mpi.h>
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.25 `Mpi_buffer_size_send.cpp` File Reference

```
#include "../inc/stdafx.h"
#include <mpi.h>
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.26 `Mpi_buffer_unpk.cpp` File Reference

```
#include "../inc/stdafx.h"
#include <mpi.h>
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.27 `MpiManager.cpp` File Reference

```
#include "../inc/stdafx.h"
#include <mpi.h>
#include "../inc/MpiManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.28 `MpiManager.h` File Reference

```
#include "stdafx.h"
```


Classes

- class [MpiManager](#)
MPI Manager class.
- struct [MpiManager::phdf5_struct](#)
Structure for storing halo information for HDF5.
- struct [MpiManager::layer_edges](#)
Structure containing global positions of the edges of halos.
- struct [MpiManager::buffer_struct](#)
Structure storing buffers sizes in each direction for particular grid.

Macros

- `#define range_i_left i = 0; i < GridUtils::downToLimit((int)pow(2, g->level + 1), N_lim); i++`
For loop definition for left halo.
- `#define range_j_down j = 0; j < GridUtils::downToLimit((int)pow(2, g->level + 1), M_lim); j++`
For loop definition for bottom halo.
- `#define range_k_front k = 0; k < GridUtils::downToLimit((int)pow(2, g->level + 1), K_lim); k++`
For loop definition for front halo.
- `#define range_i_right i = GridUtils::upToZero(N_lim - (int)pow(2, g->level + 1)); i < N_lim; i++`
For loop definition for right halo.
- `#define range_j_up j = GridUtils::upToZero(M_lim - (int)pow(2, g->level + 1)); j < M_lim; j++`
For loop definition for top halo.
- `#define range_k_back k = GridUtils::upToZero(K_lim - (int)pow(2, g->level + 1)); k < K_lim; k++`
For loop definition for back halo.

6.28.1 Macro Definition Documentation

6.28.1.1 `#define range_i_left i = 0; i < GridUtils::downToLimit((int)pow(2, g->level + 1), N_lim); i++`

For loop definition for left halo.

6.28.1.2 `#define range_i_right i = GridUtils::upToZero(N_lim - (int)pow(2, g->level + 1)); i < N_lim; i++`

For loop definition for right halo.

6.28.1.3 `#define range_j_down j = 0; j < GridUtils::downToLimit((int)pow(2, g->level + 1), M_lim); j++`

For loop definition for bottom halo.

6.28.1.4 `#define range_j_up j = GridUtils::upToZero(M_lim - (int)pow(2, g->level + 1)); j < M_lim; j++`

For loop definition for top halo.

6.28.1.5 `#define range_k_back k = GridUtils::upToZero(K_lim - (int)pow(2, g->level + 1)); k < K_lim; k++`

For loop definition for back halo.

6.28.1.6 `#define range_k_front k = 0; k < GridUtils::downToLimit((int)pow(2, g->level + 1), K_lim); k++`

For loop definition for front halo.

6.29 ObjectManager.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/ObjectManager.h"
#include "../inc/GridObj.h"
#include "../inc/GridUtils.h"
```

6.30 ObjectManager.h File Reference

```
#include "stdafx.h"
#include "IVector.h"
#include "IBMarker.h"
#include "IBBody.h"
#include "BFLBody.h"
```

Classes

- class [ObjectManager](#)
Object Manager class.

Enumerations

- enum [eObjectType](#) { [eBBBCloud](#), [eBFLCloud](#), [eIBBCloud](#) }
Specifies the type of body being processed.

6.30.1 Enumeration Type Documentation

6.30.1.1 enum eObjectType

Specifies the type of body being processed.

Enumerator

- eBBBCloud** Bounce-back body.
- eBFLCloud** BFL body.
- eIBBCloud** Immersed boundary body.

6.31 ObjectManager_init_bflbody.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/ObjectManager.h"
```

6.32 ObjectManager_init_ibmbody.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/ObjectManager.h"
```

6.33 ObjectManager_ops_ibm.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/ObjectManager.h"
#include "../inc/MpiManager.h"
#include "../inc/GridUtils.h"
```

6.34 ObjectManager_ops_ibmflex.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/GridObj.h"
#include "../inc/ObjectManager.h"
#include "../inc/MpiManager.h"
```

Macros

- `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`
- `#define TINY 1.0e-20`
- `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`

6.34.1 Macro Definition Documentation

6.34.1.1 `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`

6.34.1.2 `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`

6.34.1.3 `#define TINY 1.0e-20`

6.35 ObjectManager_ops_io.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/ObjectManager.h"
#include "../inc/GridUtils.h"
#include "../inc/PCpts.h"
#include "../inc/GridObj.h"
#include "../inc/MpiManager.h"
```

6.36 PCpts.h File Reference

```
#include "stdafx.h"
```

Classes

- class [PCpts](#)
Class to hold point cloud data.

6.37 stdafx.cpp File Reference

```
#include "../inc/stdafx.h"
```

Variables

- const int [c](#) [3][[L_nVels](#)]
Lattice velocities.
- const double [w](#) [[L_nVels](#)]
Quadrature weights.
- const double [cs](#) = 1.0 / sqrt(3.0)
Lattice sound speed.

6.37.1 Variable Documentation

6.37.1.1 const int c[3][L_nVels]

Initial value:

```
=
{
    { 1, -1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 1, -1, -1, 1, -1, 1, 1, -1, 0 },
    { 0, 0, 1, -1, 0, 0, 1, -1, 1, -1, 0, 0, 0, 0, 1, -1, -1, 1, 1, -1, -1, 1, 1, -1, 0 },
    { 0, 0, 0, 0, 1, -1, 1, -1, -1, 1, 1, -1, -1, 1, 0, 0, 0, 0, 1, -1, 1, -1, 1, -1, 0 }
}
```

Lattice velocities.

6.37.1.2 const double cs = 1.0 / sqrt(3.0)

Lattice sound speed.

6.37.1.3 const double w[L_nVels]

Initial value:

```
=
{ 2.0 / 27.0, 2.0 / 27.0, 2.0 / 27.0, 2.0 / 27.0, 2.0 / 27.0, 2.0 / 27.0,
  1.0 / 54.0, 1.0 / 54.0, 1.0 / 54.0, 1.0 / 54.0, 1.0 / 54.0, 1.0 / 54.0,
    1.0 / 54.0, 1.0 / 54.0, 1.0 / 54.0,
  1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0,
    1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0, 1.0 / 216.0,
    8.0 / 27.0 }
```

Quadrature weights.

6.38 stdafx.h File Reference

```
#include <algorithm>
#include <cmath>
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <numeric>
#include <stdlib.h>
#include <cstring>
#include <stdio.h>
#include "definitions.h"
```

Macros

- `#define LUMA_FAILED 12345`
Error definition.
- `#define L_IS_NAN std::isnan`
Not a Number declaration (Unix)

Variables

- `const int c [3][L_nVels]`
Lattice velocities.
- `const double w [L_nVels]`
Quadrature weights.
- `const double cs`
Lattice sound speed.

6.38.1 Macro Definition Documentation

6.38.1.1 #define L_IS_NAN std::isnan

Not a Number declaration (Unix)

6.38.1.2 `#define LUMA_FAILED 12345`

Error definition.

6.38.2 Variable Documentation

6.38.2.1 `const int c[3][L_nVels]`

Lattice velocities.

6.38.2.2 `const double cs`

Lattice sound speed.

6.38.2.3 `const double w[L_nVels]`

Quadrature weights.

Index

- _Owner
 - Body, [16](#)
 - ~BFLBody
 - BFLBody, [10](#)
 - ~BFLMarker
 - BFLMarker, [12](#)
 - ~Body
 - Body, [14](#)
 - ~GridObj
 - GridObj, [21](#)
 - ~IBBody
 - IBBody, [38](#)
 - ~IBMarker
 - IBMarker, [42](#)
 - ~IVector
 - IVector, [45](#)
 - ~Marker
 - Marker, [48](#)
 - ~MarkerData
 - MarkerData, [50](#)
 - ~PCpts
 - PCpts, [65](#)
- add
 - GridUtils, [32](#)
- addMarker
 - Body, [14](#)
 - IBBody, [39](#)
- BCs
 - IBBody, [41](#)
- BFLBody, [9](#)
 - ~BFLBody, [10](#)
 - BFLBody, [10](#)
 - BFLMarker, [12](#)
 - computeQ, [10](#), [11](#)
 - GridObj, [11](#)
 - Q, [11](#)
- BFLBody.cpp, [69](#)
- BFLBody.h, [69](#)
- BFLMarker, [11](#)
 - ~BFLMarker, [12](#)
 - BFLBody, [12](#)
 - BFLMarker, [12](#)
- BFLMarker.cpp, [69](#)
- BFLMarker.h, [70](#)
- bc_applyBfl
 - GridObj, [21](#)
- bc_applyBounceBack
 - GridObj, [21](#)
- bc_applyExtrapolation
 - GridObj, [21](#)
- bc_applyNrbc
 - GridObj, [22](#)
- bc_applyRegularised
 - GridObj, [22](#)
- bc_applySpecReflect
 - GridObj, [22](#)
- bc_solidSiteReset
 - GridObj, [22](#)
- bfl_build_body
 - ObjectManager, [61](#)
- Body
 - _Owner, [16](#)
 - ~Body, [14](#)
 - addMarker, [14](#)
 - Body, [14](#)
 - closed_surface, [16](#)
 - getMarkerData, [14](#)
 - isInVoxel, [15](#)
 - isVoxelMarkerVoxel, [15](#)
 - markerAdder, [15](#)
 - markers, [16](#)
 - spacing, [16](#)
- Body< MarkerType >, [13](#)
- Body.h, [70](#)
- buffer_rcv_info
 - MpiManager, [56](#)
- buffer_send_info
 - MpiManager, [56](#)
- c
 - stdafx.cpp, [96](#)
 - stdafx.h, [98](#)
- closed_surface
 - Body, [16](#)
- computeLiftDrag
 - ObjectManager, [61](#)
- computeQ
 - BFLBody, [10](#), [11](#)
- createOutputDirectory
 - GridUtils, [32](#)
- crossprod
 - GridUtils, [32](#)
- cs
 - stdafx.cpp, [96](#)
 - stdafx.h, [98](#)
- definitions.h, [70](#)
 - L_BUILD_FOR_MPI, [76](#)

[L_CHEAP_NEAREST_NODE_DETECTION, 77](#)
[L_FREESTREAM_TUNNEL, 77](#)
[L_HDF5_OUTPUT, 77](#)
[L_INLET_ON, 80](#)
[L_MPI_dir, 80](#)
[L_NumLev, 80](#)
[L_NumReg, 80](#)
[L_OUTLET_ON, 81](#)
[L_PI, 81](#)
[L_Re, 82](#)
[L_SOLID_FROM_FILE, 82](#)
[L_Timesteps, 83](#)
[L_USE_KBC_COLLISION, 83](#)
[L_Xcores, 83](#)
[L_Ycores, 84](#)
[L_Zcores, 84](#)
[L_a_x, 74](#)
[L_a_y, 74](#)
[L_a_z, 75](#)
[L_b_x, 75](#)
[L_b_y, 75](#)
[L_b_z, 75](#)
[L_bfl_length, 75](#)
[L_bfl_length_ref, 75](#)
[L_bfl_on_grid_lev, 75](#)
[L_bfl_on_grid_reg, 75](#)
[L_bfl_scale_direction, 75](#)
[L_block_on_grid_lev, 75](#)
[L_block_on_grid_reg, 76](#)
[L_block_x_max, 76](#)
[L_block_x_min, 76](#)
[L_block_y_max, 76](#)
[L_block_y_min, 76](#)
[L_block_z_max, 76](#)
[L_block_z_min, 76](#)
[L_centre_bfl_z, 76](#)
[L_centre_ibb_z, 76](#)
[L_centre_object_z, 77](#)
[L_dims, 77](#)
[L_end_BC, 77](#)
[L_grav_direction, 77](#)
[L_grav_force, 77](#)
[L_ibb_EI, 78](#)
[L_ibb_angle_horz, 77](#)
[L_ibb_angle_vert, 77](#)
[L_ibb_d, 78](#)
[L_ibb_deform, 78](#)
[L_ibb_delta_rho, 78](#)
[L_ibb_filament_length, 78](#)
[L_ibb_filament_start_x, 78](#)
[L_ibb_filament_start_y, 78](#)
[L_ibb_filament_start_z, 78](#)
[L_ibb_flex_rigid, 78](#)
[L_ibb_I, 78](#)
[L_ibb_length, 79](#)
[L_ibb_length_ref, 79](#)
[L_ibb_on_grid_lev, 79](#)
[L_ibb_on_grid_reg, 79](#)
[L_ibb_r, 79](#)
[L_ibb_scale_direction, 79](#)
[L_ibb_w, 79](#)
[L_ibb_x, 79](#)
[L_ibb_y, 79](#)
[L_ibb_z, 79](#)
[L_K, 80](#)
[L_M, 80](#)
[L_N, 80](#)
[L_nVels, 80](#)
[L_num_markers, 80](#)
[L_object_length, 80](#)
[L_object_length_ref, 81](#)
[L_object_on_grid_lev, 81](#)
[L_object_on_grid_reg, 81](#)
[L_object_scale_direction, 81](#)
[L_out_every, 81](#)
[L_out_every_forces, 81](#)
[L_out_every_probe, 81](#)
[L_output_precision, 81](#)
[L_restart_out_every, 82](#)
[L_rho_in, 82](#)
[L_start_BC, 82](#)
[L_start_bfl_x, 82](#)
[L_start_bfl_y, 82](#)
[L_start_ibb_x, 82](#)
[L_start_ibb_y, 82](#)
[L_start_object_x, 82](#)
[L_start_object_y, 83](#)
[L_u_0x, 83](#)
[L_u_0y, 83](#)
[L_u_0z, 83](#)
[L_u_max, 83](#)
[L_u_ref, 83](#)
[L_wall_thickness, 83](#)
[LUMA_VERSION, 84](#)
[nProbes, 84](#)
[RefXend, 84](#)
[RefXstart, 84](#)
[RefYend, 84](#)
[RefYstart, 84](#)
[RefZend, 84](#)
[RefZstart, 84](#)
[xProbeLims, 84](#)
[yProbeLims, 84](#)
[zProbeLims, 84](#)
 deformable
 [IBBody, 41](#)
 delta_rho
 [IBBody, 41](#)
 deltaval
 [IBMarker, 43](#)
 desired_vel
 [IBMarker, 43](#)
 destroyInstance
 [MpiManager, 54](#)
 [ObjectManager, 61](#)
 dilation

- IBMarker, 43
- dir_reflect
 - GridUtils, 36
- dotprod
 - GridUtils, 32
- downToLimit
 - GridUtils, 32
- dt
 - GridObj, 28
- eBBBCloud
 - ObjectManager.h, 94
- eBCAll
 - GridObj.h, 85
- eBCBFL
 - GridObj.h, 85
- eBCInlet
 - GridObj.h, 85
- eBCInletOutlet
 - GridObj.h, 85
- eBCOutlet
 - GridObj.h, 85
- eBCSolidSymmetry
 - GridObj.h, 85
- eBCType
 - GridObj.h, 85
- eBFLCloud
 - ObjectManager.h, 94
- eBFL
 - GridObj.h, 86
- eCartesianDirection
 - GridUtils.h, 87
- eFluid
 - GridObj.h, 86
- eHdf5SlabType
 - hdf5luma.h, 89
- eIBBCloud
 - ObjectManager.h, 94
- eInlet
 - GridObj.h, 86
- eMaximum
 - GridUtils.h, 88
- eMinMax
 - GridUtils.h, 87
- eMinimum
 - GridUtils.h, 88
- eObjectType
 - ObjectManager.h, 94
- eOutlet
 - GridObj.h, 86
- ePosX
 - hdf5luma.h, 89
- ePosY
 - hdf5luma.h, 89
- ePosZ
 - hdf5luma.h, 89
- eProductVector
 - hdf5luma.h, 89
- eRefined
 - GridObj.h, 86
- eRefinedInlet
 - GridObj.h, 86
- eRefinedSolid
 - GridObj.h, 86
- eRefinedSymmetry
 - GridObj.h, 86
- eScalar
 - hdf5luma.h, 89
- eSolid
 - GridObj.h, 86
- eSymmetry
 - GridObj.h, 86
- eTransitionToCoarser
 - GridObj.h, 86
- eTransitionToFiner
 - GridObj.h, 86
- eType
 - GridObj.h, 85
- eVector
 - hdf5luma.h, 89
- eXDirection
 - GridUtils.h, 87
- eYDirection
 - GridUtils.h, 87
- eZDirection
 - GridUtils.h, 87
- epsilon
 - IBMarker, 43
- f_buffer_recv
 - MpiManager, 56
- f_buffer_send
 - MpiManager, 56
- factorial
 - GridUtils, 32
- flex_rigid
 - IBBody, 41
 - IBMarker, 43
- flexural_rigidity
 - IBBody, 41
- fluid_vel
 - IBMarker, 43
- force_xyz
 - IBMarker, 44
- getCoarseIndices
 - GridUtils, 32
- getFineIndices
 - GridUtils, 33
- getGrid
 - GridUtils, 33
- getInstance
 - MpiManager, 54
 - ObjectManager, 62
- getMarkerData
 - Body, 14
- getOpposite
 - GridUtils, 33

- getVoxInd
 - GridUtils, 33
- global_dims
 - MpiManager, 57
- global_edge_ind
 - MpiManager, 57
- global_edge_pos
 - MpiManager, 57
- global_to_local
 - GridUtils, 33
- GridObj, 17
 - ~GridObj, 21
 - BFLBody, 11
 - bc_applyBfl, 21
 - bc_applyBounceBack, 21
 - bc_applyExtrapolation, 21
 - bc_applyNrbc, 22
 - bc_applyRegularised, 22
 - bc_applySpecReflect, 22
 - bc_solidSiteReset, 22
 - dt, 28
 - GridObj, 20
 - GridUtils, 28
 - io_hdf5, 23
 - io_lite, 23
 - io_probeOutput, 23
 - io_restart, 23
 - io_textout, 23
 - K_lim, 28
 - LBM_addSubGrid, 24
 - LBM_boundary, 24
 - LBM_coalesce, 24
 - LBM_collide, 24
 - LBM_explode, 25
 - LBM_forcegrid, 25
 - LBM_init_getInletProfile, 25
 - LBM_initBoundLab, 25
 - LBM_initGrid, 26
 - LBM_initRefinedLab, 26
 - LBM_initRho, 26
 - LBM_initSolidLab, 26
 - LBM_initSubGrid, 26
 - LBM_initVelocity, 27
 - LBM_kbcCollide, 27
 - LBM_macro, 27
 - LBM_multi, 27
 - LBM_stream, 28
 - LatTyp, 28
 - level, 28
 - M_lim, 28
 - MpiManager, 28
 - N_lim, 29
 - nu, 29
 - ObjectManager, 28, 64
 - omega, 29
 - t, 29
 - timeav_mpi_overhead, 29
 - timeav_timestep, 29
 - XInd, 29
 - XPos, 29
 - YInd, 29
 - YPos, 29
 - ZInd, 30
 - ZPos, 30
- GridObj.cpp, 85
- GridObj.h, 85
 - eBCAll, 85
 - eBCBFL, 85
 - eBCInlet, 85
 - eBCInletOutlet, 85
 - eBCOutlet, 85
 - eBCSolidSymmetry, 85
 - eBCType, 85
 - eBFL, 86
 - eFluid, 86
 - eInlet, 86
 - eOutlet, 86
 - eRefined, 86
 - eRefinedInlet, 86
 - eRefinedSolid, 86
 - eRefinedSymmetry, 86
 - eSolid, 86
 - eSymmetry, 86
 - eTransitionToCoarser, 86
 - eTransitionToFiner, 86
 - eType, 85
- GridObj_init_grids.cpp, 86
- GridObj_ops_boundary.cpp, 86
- GridObj_ops_io.cpp, 86
- GridObj_ops_lbm.cpp, 87
- GridUtils, 30
 - add, 32
 - createOutputDirectory, 32
 - crossprod, 32
 - dir_reflect, 36
 - dotprod, 32
 - downToLimit, 32
 - factorial, 32
 - getCoarseIndices, 32
 - getFineIndices, 33
 - getGrid, 33
 - getOpposite, 33
 - getVoxInd, 33
 - global_to_local, 33
 - GridObj, 28
 - hasThisSubGrid, 34
 - indexToPosition, 34
 - isOffGrid, 34
 - isOnRecvLayer, 34
 - isOnSenderLayer, 34
 - isOnThisRank, 34
 - isOverlapPeriodic, 34
 - linspace, 34
 - local_to_global, 34
 - logfile, 36
 - matrix_multiply, 35

- onespace, 35
- path_str, 37
- stridedCopy, 35
- subtract, 35
- upToZero, 35
- vecmultiply, 35
- vecnorm, 35, 36
- GridUtils.cpp, 87
- GridUtils.h, 87
 - eCartesianDirection, 87
 - eMaximum, 88
 - eMinMax, 87
 - eMinimum, 88
 - eXDirection, 87
 - eYDirection, 87
 - eZDirection, 87
- Grids
 - MpiManager, 57
- groupID
 - IBBody, 41
- H5_BUILT_AS_DYNAMIC_LIB
 - hdf5luma.h, 89
- HDF5_EXT_SZIP
 - hdf5luma.h, 89
- HDF5_EXT_ZLIB
 - hdf5luma.h, 89
- halo_max
 - MpiManager::phdf5_struct, 67
- halo_min
 - MpiManager::phdf5_struct, 67
- hasThisSubGrid
 - GridUtils, 34
- hdf5_writeDataSet
 - hdf5luma.h, 89
- hdf5luma.h, 88
 - eHdf5SlabType, 89
 - ePosX, 89
 - ePosY, 89
 - ePosZ, 89
 - eProductVector, 89
 - eScalar, 89
 - eVector, 89
 - H5_BUILT_AS_DYNAMIC_LIB, 89
 - HDF5_EXT_SZIP, 89
 - HDF5_EXT_ZLIB, 89
 - hdf5_writeDataSet, 89
- i
 - MarkerData, 51
- i_end
 - MpiManager::phdf5_struct, 67
- i_start
 - MpiManager::phdf5_struct, 67
- IBBody, 37
 - ~IBBody, 38
 - addMarker, 39
 - BCs, 41
 - deformable, 41
 - delta_rho, 41
 - flex_rigid, 41
 - flexural_rigidity, 41
 - groupID, 41
 - IBBody, 38
 - IBMarker, 43
 - makeBody, 39, 40
 - ObjectManager, 40
 - tension, 41
- IBBody.cpp, 90
- IBBody.h, 90
- IBMarker, 41
 - ~IBMarker, 42
 - deltaval, 43
 - desired_vel, 43
 - dilation, 43
 - epsilon, 43
 - flex_rigid, 43
 - fluid_vel, 43
 - force_xyz, 44
 - IBBody, 43
 - IBMarker, 42, 43
 - local_area, 44
 - ObjectManager, 43
 - position_old, 44
- IBMarker.cpp, 90
- IBMarker.h, 90
- IVector
 - ~IVector, 45
 - IVector, 45
 - operator(), 45, 46
- IVector< GenTyp >, 44
- IVector.h, 90
- ibm_apply
 - ObjectManager, 62
- ibm_banbks
 - ObjectManager, 62
- ibm_bandec
 - ObjectManager, 62
- ibm_bicgstab
 - ObjectManager, 62
- ibm_build_body
 - ObjectManager, 62
- ibm_computeforce
 - ObjectManager, 63
- ibm_deltakernel
 - ObjectManager, 63
- ibm_findepsilon
 - ObjectManager, 63
- ibm_findsupport
 - ObjectManager, 63
- ibm_initialise
 - ObjectManager, 63
- ibm_interpol
 - ObjectManager, 63
- ibm_jacowire
 - ObjectManager, 63
- ibm_move_bodies

- ObjectManager, [63](#)
- ibm_position_update
 - ObjectManager, [63](#)
- ibm_position_update_grp
 - ObjectManager, [63](#)
- ibm_spread
 - ObjectManager, [63](#)
- ID
 - MarkerData, [51](#)
- indexToPosition
 - GridUtils, [34](#)
- io_hdf5
 - GridObj, [23](#)
- io_lite
 - GridObj, [23](#)
- io_probeOutput
 - GridObj, [23](#)
- io_readInCloud
 - ObjectManager, [63](#)
- io_restart
 - GridObj, [23](#)
 - ObjectManager, [63](#)
- io_textout
 - GridObj, [23](#)
- io_vtk_IBwriter
 - ObjectManager, [64](#)
- io_write_body_pos
 - ObjectManager, [64](#)
- io_write_lift_drag
 - ObjectManager, [64](#)
- io_writeForceOnObject
 - ObjectManager, [64](#)
- isInVoxel
 - Body, [15](#)
- isOffGrid
 - GridUtils, [34](#)
- isOnRecvLayer
 - GridUtils, [34](#)
- isOnSenderLayer
 - GridUtils, [34](#)
- isOnThisRank
 - GridUtils, [34](#)
- isOverlapPeriodic
 - GridUtils, [34](#)
- isVoxelMarkerVoxel
 - Body, [15](#)
- j
 - MarkerData, [51](#)
- j_end
 - MpiManager::phdf5_struct, [67](#)
- j_start
 - MpiManager::phdf5_struct, [67](#)
- k
 - MarkerData, [51](#)
- k_end
 - MpiManager::phdf5_struct, [67](#)
- K_lim
 - GridObj, [28](#)
- k_start
 - MpiManager::phdf5_struct, [67](#)
- L_BUILD_FOR_MPI
 - definitions.h, [76](#)
- L_CHEAP_NEAREST_NODE_DETECTION
 - definitions.h, [77](#)
- L_FREESTREAM_TUNNEL
 - definitions.h, [77](#)
- L_HDF5_OUTPUT
 - definitions.h, [77](#)
- L_INLET_ON
 - definitions.h, [80](#)
- L_IS_NAN
 - stdafx.h, [97](#)
- L_MPI_dir
 - definitions.h, [80](#)
- L_NumLev
 - definitions.h, [80](#)
- L_NumReg
 - definitions.h, [80](#)
- L_OUTLET_ON
 - definitions.h, [81](#)
- L_PI
 - definitions.h, [81](#)
- L_Re
 - definitions.h, [82](#)
- L_SOLID_FROM_FILE
 - definitions.h, [82](#)
- L_Timesteps
 - definitions.h, [83](#)
- L_USE_KBC_COLLISION
 - definitions.h, [83](#)
- L_Xcores
 - definitions.h, [83](#)
- L_Ycores
 - definitions.h, [84](#)
- L_Zcores
 - definitions.h, [84](#)
- L_a_x
 - definitions.h, [74](#)
- L_a_y
 - definitions.h, [74](#)
- L_a_z
 - definitions.h, [75](#)
- L_b_x
 - definitions.h, [75](#)
- L_b_y
 - definitions.h, [75](#)
- L_b_z
 - definitions.h, [75](#)
- L_bfl_length
 - definitions.h, [75](#)
- L_bfl_length_ref
 - definitions.h, [75](#)
- L_bfl_on_grid_lev
 - definitions.h, [75](#)
- L_bfl_on_grid_reg

- definitions.h, [75](#)
- L_bfl_scale_direction
 - definitions.h, [75](#)
- L_block_on_grid_lev
 - definitions.h, [75](#)
- L_block_on_grid_reg
 - definitions.h, [76](#)
- L_block_x_max
 - definitions.h, [76](#)
- L_block_x_min
 - definitions.h, [76](#)
- L_block_y_max
 - definitions.h, [76](#)
- L_block_y_min
 - definitions.h, [76](#)
- L_block_z_max
 - definitions.h, [76](#)
- L_block_z_min
 - definitions.h, [76](#)
- L_centre_bfl_z
 - definitions.h, [76](#)
- L_centre_ibb_z
 - definitions.h, [76](#)
- L_centre_object_z
 - definitions.h, [77](#)
- L_dims
 - definitions.h, [77](#)
- L_end_BC
 - definitions.h, [77](#)
- L_grav_direction
 - definitions.h, [77](#)
- L_grav_force
 - definitions.h, [77](#)
- L_ibb_EI
 - definitions.h, [78](#)
- L_ibb_angle_horz
 - definitions.h, [77](#)
- L_ibb_angle_vert
 - definitions.h, [77](#)
- L_ibb_d
 - definitions.h, [78](#)
- L_ibb_deform
 - definitions.h, [78](#)
- L_ibb_delta_rho
 - definitions.h, [78](#)
- L_ibb_filament_length
 - definitions.h, [78](#)
- L_ibb_filament_start_x
 - definitions.h, [78](#)
- L_ibb_filament_start_y
 - definitions.h, [78](#)
- L_ibb_filament_start_z
 - definitions.h, [78](#)
- L_ibb_flex_rigid
 - definitions.h, [78](#)
- L_ibb_l
 - definitions.h, [78](#)
- L_ibb_length
 - definitions.h, [79](#)
- L_ibb_length_ref
 - definitions.h, [79](#)
- L_ibb_on_grid_lev
 - definitions.h, [79](#)
- L_ibb_on_grid_reg
 - definitions.h, [79](#)
- L_ibb_r
 - definitions.h, [79](#)
- L_ibb_scale_direction
 - definitions.h, [79](#)
- L_ibb_w
 - definitions.h, [79](#)
- L_ibb_x
 - definitions.h, [79](#)
- L_ibb_y
 - definitions.h, [79](#)
- L_ibb_z
 - definitions.h, [79](#)
- L_K
 - definitions.h, [80](#)
- L_M
 - definitions.h, [80](#)
- L_N
 - definitions.h, [80](#)
- L_nVels
 - definitions.h, [80](#)
- L_num_markers
 - definitions.h, [80](#)
- L_object_length
 - definitions.h, [80](#)
- L_object_length_ref
 - definitions.h, [81](#)
- L_object_on_grid_lev
 - definitions.h, [81](#)
- L_object_on_grid_reg
 - definitions.h, [81](#)
- L_object_scale_direction
 - definitions.h, [81](#)
- L_out_every
 - definitions.h, [81](#)
- L_out_every_forces
 - definitions.h, [81](#)
- L_out_every_probe
 - definitions.h, [81](#)
- L_output_precision
 - definitions.h, [81](#)
- L_restart_out_every
 - definitions.h, [82](#)
- L_rho_in
 - definitions.h, [82](#)
- L_start_BC
 - definitions.h, [82](#)
- L_start_bfl_x
 - definitions.h, [82](#)
- L_start_bfl_y
 - definitions.h, [82](#)
- L_start_ibb_x

- definitions.h, [82](#)
- L_start_ibb_y
 - definitions.h, [82](#)
- L_start_object_x
 - definitions.h, [82](#)
- L_start_object_y
 - definitions.h, [83](#)
- L_u_0x
 - definitions.h, [83](#)
- L_u_0y
 - definitions.h, [83](#)
- L_u_0z
 - definitions.h, [83](#)
- L_u_max
 - definitions.h, [83](#)
- L_u_ref
 - definitions.h, [83](#)
- L_wall_thickness
 - definitions.h, [83](#)
- LBM_addSubGrid
 - GridObj, [24](#)
- LBM_boundary
 - GridObj, [24](#)
- LBM_coalesce
 - GridObj, [24](#)
- LBM_collide
 - GridObj, [24](#)
- LBM_explode
 - GridObj, [25](#)
- LBM_forcegrid
 - GridObj, [25](#)
- LBM_init_getInletProfile
 - GridObj, [25](#)
- LBM_initBoundLab
 - GridObj, [25](#)
- LBM_initGrid
 - GridObj, [26](#)
- LBM_initRefinedLab
 - GridObj, [26](#)
- LBM_initRho
 - GridObj, [26](#)
- LBM_initSolidLab
 - GridObj, [26](#)
- LBM_initSubGrid
 - GridObj, [26](#)
- LBM_initVelocity
 - GridObj, [27](#)
- LBM_kbcCollide
 - GridObj, [27](#)
- LBM_macro
 - GridObj, [27](#)
- LBM_multi
 - GridObj, [27](#)
- LBM_stream
 - GridObj, [28](#)
- LUMA_FAILED
 - stdafx.h, [97](#)
- LUMA_VERSION
 - definitions.h, [84](#)
- LatTyp
 - GridObj, [28](#)
- level
 - GridObj, [28](#)
 - MpiManager::buffer_struct, [17](#)
 - MpiManager::phdf5_struct, [67](#)
- linspace
 - GridUtils, [34](#)
- local_area
 - IBMarker, [44](#)
- local_size
 - MpiManager, [57](#)
- local_to_global
 - GridUtils, [34](#)
- logfile
 - GridUtils, [36](#)
- logout
 - MpiManager, [57](#)
- M_lim
 - GridObj, [28](#)
- MPI_cartlab
 - MpiManager, [57](#)
- MPI_coords
 - MpiManager, [57](#)
- MPI_dims
 - MpiManager, [58](#)
- main
 - main_lbm.cpp, [91](#)
- main_lbm.cpp, [91](#)
- main, [91](#)
- makeBody
 - IBBody, [39, 40](#)
- Marker, [47](#)
 - ~Marker, [48](#)
 - Marker, [48](#)
 - position, [49](#)
 - supp_i, [49](#)
 - supp_j, [49](#)
 - supp_k, [49](#)
 - support_rank, [49](#)
- Marker.h, [91](#)
- markerAdder
 - Body, [15](#)
- MarkerData, [49](#)
 - ~MarkerData, [50](#)
 - i, [51](#)
 - ID, [51](#)
 - j, [51](#)
 - k, [51](#)
 - MarkerData, [50](#)
 - x, [51](#)
 - y, [51](#)
 - z, [51](#)
- markers
 - Body, [16](#)
- matrix_multiply
 - GridUtils, [35](#)

- mpi_buffer_pack
 - MpiManager, 54
- Mpi_buffer_pack.cpp, 91
- mpi_buffer_size
 - MpiManager, 54
- mpi_buffer_size_recv
 - MpiManager, 54
- Mpi_buffer_size_recv.cpp, 92
- mpi_buffer_size_send
 - MpiManager, 55
- Mpi_buffer_size_send.cpp, 92
- mpi_buffer_unpack
 - MpiManager, 55
- Mpi_buffer_unpk.cpp, 92
- mpi_buildCommunicators
 - MpiManager, 55
- mpi_communicate
 - MpiManager, 55
- mpi_getOpposite
 - MpiManager, 56
- mpi_gridbuild
 - MpiManager, 56
- mpi_init
 - MpiManager, 56
- mpi_writeout_buf
 - MpiManager, 56
- MpiManager, 51
 - buffer_recv_info, 56
 - buffer_send_info, 56
 - destroyInstance, 54
 - f_buffer_recv, 56
 - f_buffer_send, 56
 - getInstance, 54
 - global_dims, 57
 - global_edge_ind, 57
 - global_edge_pos, 57
 - GridObj, 28
 - Grids, 57
 - local_size, 57
 - logout, 57
 - MPI_cartlab, 57
 - MPI_coords, 57
 - MPI_dims, 58
 - mpi_buffer_pack, 54
 - mpi_buffer_size, 54
 - mpi_buffer_size_recv, 54
 - mpi_buffer_size_send, 55
 - mpi_buffer_unpack, 55
 - mpi_buildCommunicators, 55
 - mpi_communicate, 55
 - mpi_getOpposite, 56
 - mpi_gridbuild, 56
 - mpi_init, 56
 - mpi_writeout_buf, 56
 - my_rank, 58
 - neighbour_coords, 58
 - neighbour_rank, 58
 - num_ranks, 58
 - p_data, 58
 - recv_layer_pos, 58
 - recv_stat, 58
 - send_requests, 58
 - send_stat, 58
 - sender_layer_pos, 59
 - subGrid_comm, 59
 - world_comm, 59
- MpiManager.cpp, 92
- MpiManager.h, 92
 - range_i_left, 93
 - range_i_right, 93
 - range_j_down, 93
 - range_j_up, 93
 - range_k_back, 93
 - range_k_front, 94
- MpiManager::buffer_struct, 16
 - level, 17
 - region, 17
 - size, 17
- MpiManager::layer_edges, 46
 - X, 47
 - Y, 47
 - Z, 47
- MpiManager::phdf5_struct, 66
 - halo_max, 67
 - halo_min, 67
 - i_end, 67
 - i_start, 67
 - j_end, 67
 - j_start, 67
 - k_end, 67
 - k_start, 67
 - level, 67
 - region, 68
 - writable_data_count, 68
- my_rank
 - MpiManager, 58
- N_lim
 - GridObj, 29
- nProbes
 - definitions.h, 84
- neighbour_coords
 - MpiManager, 58
- neighbour_rank
 - MpiManager, 58
- nu
 - GridObj, 29
- num_ranks
 - MpiManager, 58
- ObjectManager, 59
 - bfl_build_body, 61
 - computeLiftDrag, 61
 - destroyInstance, 61
 - getInstance, 62
 - GridObj, 28, 64
 - IBBody, 40

- IBMarker, [43](#)
- ibm_apply, [62](#)
- ibm_banbks, [62](#)
- ibm_bandec, [62](#)
- ibm_bicgstab, [62](#)
- ibm_build_body, [62](#)
- ibm_computeforce, [63](#)
- ibm_deltakernel, [63](#)
- ibm_findepilon, [63](#)
- ibm_findsupport, [63](#)
- ibm_initialise, [63](#)
- ibm_interpol, [63](#)
- ibm_jacowire, [63](#)
- ibm_move_bodies, [63](#)
- ibm_position_update, [63](#)
- ibm_position_update_grp, [63](#)
- ibm_spread, [63](#)
- io_readInCloud, [63](#)
- io_restart, [63](#)
- io_vtk_IBwriter, [64](#)
- io_write_body_pos, [64](#)
- io_write_lift_drag, [64](#)
- io_writeForceOnObject, [64](#)
- ObjectManager.cpp, [94](#)
- ObjectManager.h, [94](#)
 - eBBBCloud, [94](#)
 - eBFLCloud, [94](#)
 - eIBBCloud, [94](#)
 - eObjectType, [94](#)
- ObjectManager_init_bflbody.cpp, [95](#)
- ObjectManager_init_ibmbody.cpp, [95](#)
- ObjectManager_ops_ibm.cpp, [95](#)
- ObjectManager_ops_ibmflex.cpp, [95](#)
 - SWAP, [95](#)
 - TINY, [95](#)
- ObjectManager_ops_io.cpp, [95](#)
- omega
 - GridObj, [29](#)
- onespace
 - GridUtils, [35](#)
- operator()
 - IVector, [45](#), [46](#)
- p_data
 - MpiManager, [58](#)
- PCpts, [65](#)
 - ~PCpts, [65](#)
 - PCpts, [65](#)
 - x, [66](#)
 - y, [66](#)
 - z, [66](#)
- PCpts.h, [96](#)
- path_str
 - GridUtils, [37](#)
- position
 - Marker, [49](#)
- position_old
 - IBMarker, [44](#)

- Q
 - BFLBody, [11](#)
- range_i_left
 - MpiManager.h, [93](#)
- range_i_right
 - MpiManager.h, [93](#)
- range_j_down
 - MpiManager.h, [93](#)
- range_j_up
 - MpiManager.h, [93](#)
- range_k_back
 - MpiManager.h, [93](#)
- range_k_front
 - MpiManager.h, [94](#)
- recv_layer_pos
 - MpiManager, [58](#)
- recv_stat
 - MpiManager, [58](#)
- RefXend
 - definitions.h, [84](#)
- RefXstart
 - definitions.h, [84](#)
- RefYend
 - definitions.h, [84](#)
- RefYstart
 - definitions.h, [84](#)
- RefZend
 - definitions.h, [84](#)
- RefZstart
 - definitions.h, [84](#)
- region
 - MpiManager::buffer_struct, [17](#)
 - MpiManager::phdf5_struct, [68](#)
- SWAP
 - ObjectManager_ops_ibmflex.cpp, [95](#)
- send_requests
 - MpiManager, [58](#)
- send_stat
 - MpiManager, [58](#)
- sender_layer_pos
 - MpiManager, [59](#)
- size
 - MpiManager::buffer_struct, [17](#)
- spacing
 - Body, [16](#)
- stdafx.cpp, [96](#)
 - c, [96](#)
 - cs, [96](#)
 - w, [96](#)
- stdafx.h, [97](#)
 - c, [98](#)
 - cs, [98](#)
 - L_IS_NAN, [97](#)
 - LUMA_FAILED, [97](#)
 - w, [98](#)
- stridedCopy
 - GridUtils, [35](#)

- subGrid_comm
 - MpiManager, [59](#)
- subtract
 - GridUtils, [35](#)
- supp_i
 - Marker, [49](#)
- supp_j
 - Marker, [49](#)
- supp_k
 - Marker, [49](#)
- support_rank
 - Marker, [49](#)
- t
 - GridObj, [29](#)
- TINY
 - ObjectManager_ops_ibmflex.cpp, [95](#)
- tension
 - IBBody, [41](#)
- timeav_mpi_overhead
 - GridObj, [29](#)
- timeav_timestep
 - GridObj, [29](#)
- upToZero
 - GridUtils, [35](#)
- vecmultiply
 - GridUtils, [35](#)
- vecnorm
 - GridUtils, [35](#), [36](#)
- w
 - stdafx.cpp, [96](#)
 - stdafx.h, [98](#)
- world_comm
 - MpiManager, [59](#)
- writable_data_count
 - MpiManager::phdf5_struct, [68](#)
- X
 - MpiManager::layer_edges, [47](#)
- x
 - MarkerData, [51](#)
 - PCpts, [66](#)
- XInd
 - GridObj, [29](#)
- XPos
 - GridObj, [29](#)
- xProbeLims
 - definitions.h, [84](#)
- Y
 - MpiManager::layer_edges, [47](#)
- y
 - MarkerData, [51](#)
 - PCpts, [66](#)
- YInd
 - GridObj, [29](#)
- YPos
 - GridObj, [29](#)
- yProbeLims
 - definitions.h, [84](#)
- Z
 - MpiManager::layer_edges, [47](#)
- z
 - MarkerData, [51](#)
 - PCpts, [66](#)
- ZInd
 - GridObj, [30](#)
- ZPos
 - GridObj, [30](#)
- zProbeLims
 - definitions.h, [84](#)