

LUMA

1.4.0-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(GridObj *g, size_t id, PCpts *_PCpts)	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, GridObj const *const body_owner)	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, size_t id)	14
5.3.2.4	Body(GridObj *g, size_t id, PCpts *_PCpts)	14
5.3.3	Member Function Documentation	14
5.3.3.1	addMarker(double x, double y, double z)	14
5.3.3.2	buildFromCloud(PCpts *_PCpts)	15
5.3.3.3	getMarkerData(double x, double y, double z)	15
5.3.3.4	passToVoxelFilter(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	id	16
5.3.4.4	markers	16
5.3.4.5	spacing	16
5.4	MpiManager::buffer_struct Struct Reference	16
5.4.1	Detailed Description	16
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	GridManager Class Reference	17
5.5.1	Detailed Description	18
5.5.2	Member Function Documentation	18
5.5.2.1	createWritableDataStore(HDFstruct *&datastruct)	18
5.5.2.2	createWritableDataStore(GridObj const *const targetGrid)	18
5.5.2.3	destroyInstance()	19
5.5.2.4	getInstance()	19
5.5.2.5	setLocalCoarseSize(const std::vector< int > &size_vector)	19
5.5.3	Friends And Related Function Documentation	19
5.5.3.1	GridObj	19
5.5.3.2	GridUtils	19
5.5.3.3	MpiManager	19
5.5.4	Member Data Documentation	19
5.5.4.1	global_edges	19
5.5.4.2	global_size	19
5.5.4.3	Grids	20
5.5.4.4	local_size	20
5.5.4.5	p_data	20
5.5.4.6	subgrid_tlayer_key	20
5.6	GridObj Class Reference	20
5.6.1	Detailed Description	22
5.6.2	Constructor & Destructor Documentation	23
5.6.2.1	GridObj()	23
5.6.2.2	GridObj(int level)	23
5.6.2.3	GridObj(int RegionNumber, GridObj &pGrid)	23
5.6.2.4	~GridObj()	23
5.6.3	Member Function Documentation	23
5.6.3.1	bc_applyBfl(int i, int j, int k)	23

5.6.3.2	bc_applyBounceBack(int label, int i, int j, int k)	24
5.6.3.3	bc_applyExtrapolation(int label, int i, int j, int k)	24
5.6.3.4	bc_applyNrbc(int i, int j, int k)	24
5.6.3.5	bc_applyRegularised(int label, int i, int j, int k)	24
5.6.3.6	bc_applySpecReflect(int label, int i, int j, int k)	25
5.6.3.7	io_fgaout()	25
5.6.3.8	io_hdf5(double tval)	25
5.6.3.9	io_lite(double tval, std::string Tag)	25
5.6.3.10	io_probeOutput()	26
5.6.3.11	io_restart(eIOFlag IO_flag)	26
5.6.3.12	io_textout(std::string output_tag)	26
5.6.3.13	LBM_addSubGrid(int RegionNumber)	26
5.6.3.14	LBM_init_getInletProfile()	26
5.6.3.15	LBM_initBoundLab()	27
5.6.3.16	LBM_initGrid()	27
5.6.3.17	LBM_initGridToGridMappings(GridObj &pGrid)	27
5.6.3.18	LBM_initPositionVector(double start_pos, double end_pos, eCartesianDirection dir)	27
5.6.3.19	LBM_initRefinedLab(GridObj &pGrid)	27
5.6.3.20	LBM_initRho()	28
5.6.3.21	LBM_initSolidLab()	28
5.6.3.22	LBM_initSubGrid(GridObj &pGrid)	28
5.6.3.23	LBM_initVelocity()	28
5.6.3.24	LBM_kbcCollide(int i, int j, int k, IVector< double > &f_new)	28
5.6.3.25	LBM_macro(int i, int j, int k)	28
5.6.3.26	LBM_multi_opt(int subcycle=0)	29
5.6.3.27	LBM_resetForces()	29
5.6.4	Friends And Related Function Documentation	29
5.6.4.1	GridUtils	29
5.6.4.2	MpiManager	29
5.6.4.3	ObjectManager	29

5.6.5	Member Data Documentation	29
5.6.5.1	dh	29
5.6.5.2	dt	29
5.6.5.3	K_lim	29
5.6.5.4	LatTyp	30
5.6.5.5	level	30
5.6.5.6	M_lim	30
5.6.5.7	N_lim	30
5.6.5.8	nu	30
5.6.5.9	omega	30
5.6.5.10	region_number	30
5.6.5.11	t	30
5.6.5.12	timeav_mpi_overhead	30
5.6.5.13	timeav_timestep	30
5.6.5.14	XOrigin	31
5.6.5.15	XPos	31
5.6.5.16	YOrigin	31
5.6.5.17	YPos	31
5.6.5.18	ZOrigin	31
5.6.5.19	ZPos	31
5.7	GridUnits Class Reference	31
5.7.1	Detailed Description	32
5.7.2	Constructor & Destructor Documentation	32
5.7.2.1	GridUnits()	32
5.7.2.2	~GridUnits()	32
5.7.3	Member Function Documentation	32
5.7.3.1	m2cm(const T meters)	32
5.7.3.2	ulat2uphys(T ulat, GridObj *currentGrid)	32
5.8	GridUtils Class Reference	33
5.8.1	Detailed Description	35

5.8.2	Member Function Documentation	35
5.8.2.1	<code>add(std::vector< double > a, std::vector< double > b)</code>	35
5.8.2.2	<code>createOutputDirectory(std::string path_str)</code>	35
5.8.2.3	<code>crossprod(std::vector< double > vec1, std::vector< double > vec2)</code>	36
5.8.2.4	<code>dotprod(std::vector< double > vec1, std::vector< double > vec2)</code>	36
5.8.2.5	<code>downToLimit(NumType x, NumType limit)</code>	36
5.8.2.6	<code>factorial(NumType n)</code>	36
5.8.2.7	<code>getCoarseIndices(int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)</code>	37
5.8.2.8	<code>getEnclosingVoxel(double x, double y, double z, GridObj const *const g, std::vector< int > *ijk)</code>	37
5.8.2.9	<code>getEnclosingVoxel(double x, GridObj const *const g, eCartesianDirection dir, int *ijk)</code>	38
5.8.2.10	<code>getFineIndices(int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)</code>	38
5.8.2.11	<code>getGrid(GridObj *const Grids, int level, int region, GridObj *&ptr)</code>	38
5.8.2.12	<code>getMpiDirection(int offset_vector[])</code>	39
5.8.2.13	<code>getOpposite(int direction)</code>	39
5.8.2.14	<code>intersectsRefinedRegion(GridObj const &pGrid, int RegNum)</code>	39
5.8.2.15	<code>isOffGrid(int i, int j, int k, GridObj const *const g)</code>	40
5.8.2.16	<code>isOnRecvLayer(double pos_x, double pos_y, double pos_z)</code>	40
5.8.2.17	<code>isOnRecvLayer(double site_position, eCartMinMax edge)</code>	40
5.8.2.18	<code>isOnSenderLayer(double pos_x, double pos_y, double pos_z)</code>	41
5.8.2.19	<code>isOnSenderLayer(double site_position, eCartMinMax edge)</code>	41
5.8.2.20	<code>isOnThisRank(double x, double y, double z, eLocationOnRank *loc=nullptr, GridObj const *const grid=nullptr, std::vector< int > *pos=nullptr)</code>	41
5.8.2.21	<code>isOnThisRank(double xyz, eCartesianDirection dir, eLocationOnRank *loc=nullptr, GridObj const *const grid=nullptr, int *pos=nullptr)</code>	42
5.8.2.22	<code>isOnTransitionLayer(double pos_x, double pos_y, double pos_z, GridObj const *const grid)</code>	42
5.8.2.23	<code>isOnTransitionLayer(double position, eCartMinMax edge, GridObj const *const grid)</code>	43
5.8.2.24	<code>isOverlapPeriodic(int i, int j, int k, GridObj const &pGrid)</code>	43
5.8.2.25	<code>linspace(double min, double max, int n)</code>	43

5.8.2.26	<code>matrix_multiply(const std::vector< std::vector< double > > &A, const std::vector< double > &x)</code>	44
5.8.2.27	<code>onespace(int min, int max)</code>	44
5.8.2.28	<code>safeGetRank()</code>	44
5.8.2.29	<code>stridedCopy(NumType *dest, NumType *src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset=0)</code>	45
5.8.2.30	<code>subtract(std::vector< double > a, std::vector< double > b)</code>	45
5.8.2.31	<code>upToZero(NumType x)</code>	45
5.8.2.32	<code>vecmultiply(double scalar, std::vector< double > vec)</code>	45
5.8.2.33	<code>vecnorm(double vec[L_DIMS])</code>	46
5.8.2.34	<code>vecnorm(double val1, double val2)</code>	46
5.8.2.35	<code>vecnorm(double val1, double val2, double val3)</code>	46
5.8.2.36	<code>vecnorm(std::vector< double > vec)</code>	47
5.8.2.37	<code>vecnorm(NumType a1, NumType a2, NumType a3)</code>	47
5.8.2.38	<code>vecnorm(NumType a1, NumType a2)</code>	47
5.8.3	Member Data Documentation	48
5.8.3.1	<code>dir_reflect</code>	48
5.8.3.2	<code>logfile</code>	48
5.8.3.3	<code>path_str</code>	48
5.9	HDFstruct Struct Reference	48
5.9.1	Detailed Description	49
5.9.2	Member Data Documentation	49
5.9.2.1	<code>i_end</code>	49
5.9.2.2	<code>i_start</code>	49
5.9.2.3	<code>j_end</code>	49
5.9.2.4	<code>j_start</code>	49
5.9.2.5	<code>k_end</code>	49
5.9.2.6	<code>k_start</code>	49
5.9.2.7	<code>level</code>	49
5.9.2.8	<code>region</code>	49
5.9.2.9	<code>writable_data_count</code>	50

5.10	IBody Class Reference	50
5.10.1	Detailed Description	51
5.10.2	Constructor & Destructor Documentation	51
5.10.2.1	IBody(void)	51
5.10.2.2	~IBody(void)	51
5.10.2.3	IBody(GridObj *g, size_t id)	51
5.10.2.4	IBody(GridObj *g, size_t id, PCpts *_PCpts)	52
5.10.3	Member Function Documentation	52
5.10.3.1	addMarker(double x, double y, double z, bool isFlexible)	52
5.10.3.2	makeBody(double radius, std::vector< double > centre, bool isFlexible, bool isMovable, int group)	52
5.10.3.3	makeBody(std::vector< double > width_length_depth, std::vector< double > angles, std::vector< double > centre, bool isFlexible, bool isMovable, int group)	53
5.10.3.4	makeBody(int numbermarkers, std::vector< double > start_point, double fil_length, std::vector< double > angles, std::vector< int > BCs, bool isFlexible, bool isMovable, int group)	53
5.10.3.5	makeBody(std::vector< double > width_length, double angle, std::vector< double > centre, bool isFlexible, bool isMovable, int group, bool plate)	53
5.10.4	Friends And Related Function Documentation	54
5.10.4.1	IBInfo	54
5.10.4.2	ObjectManager	54
5.10.5	Member Data Documentation	54
5.10.5.1	BCs	54
5.10.5.2	delta_rho	54
5.10.5.3	flexural_rigidity	54
5.10.5.4	groupID	54
5.10.5.5	isFlexible	54
5.10.5.6	isMovable	54
5.10.5.7	tension	54
5.11	IBInfo Class Reference	55
5.11.1	Detailed Description	55
5.11.2	Constructor & Destructor Documentation	55
5.11.2.1	IBInfo()	55

5.11.2.2	IBInfo(IBBody *iBody, elBInfoType type)	55
5.11.3	Member Function Documentation	55
5.11.3.1	mapToMpiStruct(elBInfoType type)	55
5.12	IBMarker Class Reference	56
5.12.1	Detailed Description	57
5.12.2	Constructor & Destructor Documentation	57
5.12.2.1	IBMarker(void)	57
5.12.2.2	~IBMarker(void)	57
5.12.2.3	IBMarker(double xPos, double yPos, double zPos, GridObj const *const body_↔ owner, bool isFlexible=false)	57
5.12.3	Friends And Related Function Documentation	57
5.12.3.1	IBBody	57
5.12.3.2	IBInfo	57
5.12.3.3	ObjectManager	58
5.12.4	Member Data Documentation	58
5.12.4.1	deltaval	58
5.12.4.2	desired_vel	58
5.12.4.3	dilation	58
5.12.4.4	epsilon	58
5.12.4.5	fluid_vel	58
5.12.4.6	force_xyz	58
5.12.4.7	isFlexible	58
5.12.4.8	local_area	58
5.12.4.9	position_old	58
5.13	IVector< GenTyp > Class Template Reference	59
5.13.1	Detailed Description	59
5.13.2	Constructor & Destructor Documentation	59
5.13.2.1	IVector()	59
5.13.2.2	~IVector()	59
5.13.2.3	IVector(size_t size, GenTyp val)	59
5.13.3	Member Function Documentation	60

5.13.3.1	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)	60
5.13.3.2	operator()(size_t i, size_t j, size_t k, size_t j_max, size_t k_max)	60
5.13.3.3	operator()(size_t i, size_t j, size_t j_max)	61
5.14	MpiManager::layer_edges Struct Reference	61
5.14.1	Detailed Description	61
5.14.2	Member Data Documentation	61
5.14.2.1	X	61
5.14.2.2	Y	62
5.14.2.3	Z	62
5.15	Marker Class Reference	62
5.15.1	Detailed Description	63
5.15.2	Constructor & Destructor Documentation	63
5.15.2.1	Marker(void)	63
5.15.2.2	~Marker(void)	63
5.15.2.3	Marker(double x, double y, double z, GridObj const *const body_owner)	63
5.15.3	Member Data Documentation	63
5.15.3.1	position	63
5.15.3.2	supp_i	63
5.15.3.3	supp_j	63
5.15.3.4	supp_k	64
5.15.3.5	support_rank	64
5.16	MarkerData Class Reference	64
5.16.1	Detailed Description	64
5.16.2	Constructor & Destructor Documentation	64
5.16.2.1	MarkerData(int i, int j, int k, double x, double y, double z, int ID)	64
5.16.2.2	MarkerData(void)	65
5.16.2.3	~MarkerData(void)	65
5.16.3	Member Data Documentation	65
5.16.3.1	i	65
5.16.3.2	ID	65

5.16.3.3	j	65
5.16.3.4	k	65
5.16.3.5	x	65
5.16.3.6	y	66
5.16.3.7	z	66
5.17	MpiManager Class Reference	66
5.17.1	Detailed Description	68
5.17.2	Member Function Documentation	68
5.17.2.1	destroyInstance()	68
5.17.2.2	getInstance()	68
5.17.2.3	mpi_buffer_pack(int dir, GridObj *const g)	68
5.17.2.4	mpi_buffer_size()	68
5.17.2.5	mpi_buffer_size_recv(GridObj *const g)	68
5.17.2.6	mpi_buffer_size_send(GridObj *const g)	69
5.17.2.7	mpi_buffer_unpack(int dir, GridObj *const g)	69
5.17.2.8	mpi_buildCommunicators(GridManager *const grid_man)	69
5.17.2.9	mpi_communicate(int level, int regnum)	69
5.17.2.10	mpi_getOpposite(int direction)	70
5.17.2.11	mpi_gridbuild(GridManager *const grid_man)	70
5.17.2.12	mpi_init()	70
5.17.2.13	mpi_updateLoadInfo(GridManager *const grid_man)	70
5.17.2.14	mpi_writeout_buf(std::string filename, int dir)	71
5.17.3	Member Data Documentation	71
5.17.3.1	buffer_recv_info	71
5.17.3.2	buffer_send_info	71
5.17.3.3	cRankSizeX	71
5.17.3.4	cRankSizeY	71
5.17.3.5	cRankSizeZ	71
5.17.3.6	dimensions	71
5.17.3.7	f_buffer_recv	71

5.17.3.8	<code>f_buffer_send</code>	71
5.17.3.9	<code>logout</code>	72
5.17.3.10	<code>my_rank</code>	72
5.17.3.11	<code>neighbour_coords</code>	72
5.17.3.12	<code>neighbour_rank</code>	72
5.17.3.13	<code>neighbour_vectors</code>	72
5.17.3.14	<code>num_ranks</code>	72
5.17.3.15	<code>rank_coords</code>	72
5.17.3.16	<code>rank_core_edge</code>	72
5.17.3.17	<code>recv_layer_pos</code>	73
5.17.3.18	<code>recv_stat</code>	73
5.17.3.19	<code>send_requests</code>	73
5.17.3.20	<code>send_stat</code>	73
5.17.3.21	<code>sender_layer_pos</code>	73
5.17.3.22	<code>subGrid_comm</code>	73
5.17.3.23	<code>world_comm</code>	73
5.18	ObjectManager Class Reference	73
5.18.1	Detailed Description	75
5.18.2	Member Function Documentation	75
5.18.2.1	<code>bfl_buildBody(int body_type)</code>	75
5.18.2.2	<code>bfl_buildBody(PCpts *_PCpts)</code>	75
5.18.2.3	<code>computeLiftDrag(int i, int j, int k, GridObj *g)</code>	76
5.18.2.4	<code>destroyInstance()</code>	76
5.18.2.5	<code>getInstance()</code>	76
5.18.2.6	<code>getInstance(GridObj *g)</code>	76
5.18.2.7	<code>ibm_apply()</code>	76
5.18.2.8	<code>ibm_banbks(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double b[])</code>	76
5.18.2.9	<code>ibm_bandec(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double *d)</code>	77
5.18.2.10	<code>ibm_bicgstab(std::vector< std::vector< double > > &Amatrix, std::vector< double > &bVector, std::vector< double > &epsilon, double tolerance, int maxiterations)</code>	77

5.18.2.11	<code>ibm_buildBody(int body_type)</code>	78
5.18.2.12	<code>ibm_buildBody(PCpts *_PCpts, GridObj *owner)</code>	78
5.18.2.13	<code>ibm_computeForce(int ib)</code>	78
5.18.2.14	<code>ibm_deltaKernel(double rad, double dilation)</code>	78
5.18.2.15	<code>ibm_findEpsilon(int ib)</code>	79
5.18.2.16	<code>ibm_findSupport(int ib, int m)</code>	79
5.18.2.17	<code>ibm_initialise()</code>	79
5.18.2.18	<code>ibm_initialiseSupport(int ib, int m, int s, double estimated_position[])</code>	79
5.18.2.19	<code>ibm_interpol(int ib)</code>	80
5.18.2.20	<code>ibm_jacowire(int ib)</code>	80
5.18.2.21	<code>ibm_moveBodies()</code>	80
5.18.2.22	<code>ibm_positionUpdate(int ib)</code>	80
5.18.2.23	<code>ibm_positionUpdateGroup(int group)</code>	80
5.18.2.24	<code>ibm_spread(int ib)</code>	81
5.18.2.25	<code>io_readInCloud(PCpts *_PCpts, eObjectType objtype)</code>	81
5.18.2.26	<code>io_restart(eIOFlag IO_flag, int level)</code>	81
5.18.2.27	<code>io_vtkIBBWriter(double tval)</code>	81
5.18.2.28	<code>io_writeBodyPosition(int timestep)</code>	81
5.18.2.29	<code>io_writeForceOnObject(double tval)</code>	82
5.18.2.30	<code>io_writeLiftDrag(int timestep)</code>	82
5.18.3	Friends And Related Function Documentation	82
5.18.3.1	<code>GridObj</code>	82
5.19	PCpts Class Reference	82
5.19.1	Detailed Description	83
5.19.2	Constructor & Destructor Documentation	83
5.19.2.1	<code>PCpts(void)</code>	83
5.19.2.2	<code>~PCpts(void)</code>	83
5.19.3	Member Data Documentation	83
5.19.3.1	<code>x</code>	83
5.19.3.2	<code>y</code>	83
5.19.3.3	<code>z</code>	83

6 File Documentation	85
6.1 BFLBody.cpp File Reference	85
6.2 BFLBody.h File Reference	85
6.3 BFLMarker.cpp File Reference	85
6.4 BFLMarker.h File Reference	85
6.5 Body.h File Reference	86
6.6 definitions.h File Reference	86
6.6.1 Macro Definition Documentation	91
6.6.1.1 L_BFL_LENGTH	91
6.6.1.2 L_BFL_ON_GRID_LEV	91
6.6.1.3 L_BFL_ON_GRID_REG	91
6.6.1.4 L_BFL_REF_LENGTH	91
6.6.1.5 L_BFL_SCALE_DIRECTION	92
6.6.1.6 L_BLOCK_MAX_X	92
6.6.1.7 L_BLOCK_MAX_Y	92
6.6.1.8 L_BLOCK_MAX_Z	92
6.6.1.9 L_BLOCK_MAX_Z	92
6.6.1.10 L_BLOCK_MIN_X	92
6.6.1.11 L_BLOCK_MIN_Y	92
6.6.1.12 L_BLOCK_MIN_Z	92
6.6.1.13 L_BLOCK_MIN_Z	92
6.6.1.14 L_BLOCK_ON_GRID_LEV	92
6.6.1.15 L_BLOCK_ON_GRID_REG	93
6.6.1.16 L_BUILD_FOR_MPI	93
6.6.1.17 L_BX	93
6.6.1.18 L_BY	93
6.6.1.19 L_BZ	93
6.6.1.20 L_BZ	93
6.6.1.21 L_CENTRE_BFL_Z	93
6.6.1.22 L_CENTRE_BFL_Z	93

6.6.1.23	L_CENTRE_IBB_Z	93
6.6.1.24	L_CENTRE_IBB_Z	93
6.6.1.25	L_CENTRE_OBJECT_Z	94
6.6.1.26	L_CENTRE_OBJECT_Z	94
6.6.1.27	L_CLOUD_DEBUG	94
6.6.1.28	L_CSMAG	94
6.6.1.29	L_DIMS	94
6.6.1.30	L_FILAMENT_END_BC	94
6.6.1.31	L_FILAMENT_START_BC	94
6.6.1.32	L_FREESTREAM_TUNNEL	94
6.6.1.33	L_GRAVITY_DIRECTION	94
6.6.1.34	L_GRAVITY_FORCE	94
6.6.1.35	L_HDF5_OUTPUT	94
6.6.1.36	L_HDF_DEBUG	95
6.6.1.37	L_IB_ON_LEV	95
6.6.1.38	L_IB_ON_REG	95
6.6.1.39	L_IBB_ANGLE_HORZ	95
6.6.1.40	L_IBB_ANGLE_VERT	95
6.6.1.41	L_IBB_D	95
6.6.1.42	L_IBB_D	95
6.6.1.43	L_IBB_DELTA_RHO	95
6.6.1.44	L_IBB_EI	95
6.6.1.45	L_IBB_FILAMENT_LENGTH	95
6.6.1.46	L_IBB_FILAMENT_START_X	96
6.6.1.47	L_IBB_FILAMENT_START_Y	96
6.6.1.48	L_IBB_FILAMENT_START_Z	96
6.6.1.49	L_IBB_FLEXIBLE	96
6.6.1.50	L_IBB_FROM_FILE	96
6.6.1.51	L_IBB_L	96
6.6.1.52	L_IBB_LENGTH	96

6.6.1.53	L_IBB_MOVABLE	96
6.6.1.54	L_IBB_ON_GRID_LEV	96
6.6.1.55	L_IBB_ON_GRID_REG	96
6.6.1.56	L_IBB_R	97
6.6.1.57	L_IBB_REF_LENGTH	97
6.6.1.58	L_IBB_SCALE_DIRECTION	97
6.6.1.59	L_IBB_W	97
6.6.1.60	L_IBB_X	97
6.6.1.61	L_IBB_Y	97
6.6.1.62	L_IBB_Z	97
6.6.1.63	L_INIT_VERBOSE	97
6.6.1.64	L_INLET_ON	97
6.6.1.65	L_K	97
6.6.1.66	L_K	97
6.6.1.67	L_LD_OUT	97
6.6.1.68	L_LOG_TIMINGS	98
6.6.1.69	L_M	98
6.6.1.70	L_MPI_DIRS	98
6.6.1.71	L_MPI_VERBOSE	98
6.6.1.72	L_MPI_WRITE_LOAD_BALANCE	98
6.6.1.73	L_MPI_XCORES	98
6.6.1.74	L_MPI_YCORES	98
6.6.1.75	L_MPI_ZCORES	98
6.6.1.76	L_MPI_ZCORES	98
6.6.1.77	L_N	98
6.6.1.78	L_NUM_LEVELS	98
6.6.1.79	L_NUM_MARKERS	98
6.6.1.80	L_NUM_REGIONS	99
6.6.1.81	L_NUM_VELS	99
6.6.1.82	L_OBJECT_LENGTH	99

6.6.1.83	L_OBJECT_ON_GRID_LEV	99
6.6.1.84	L_OBJECT_ON_GRID_REG	99
6.6.1.85	L_OBJECT_REF_LENGTH	99
6.6.1.86	L_OBJECT_SCALE_DIRECTION	99
6.6.1.87	L_OUT_EVERY	99
6.6.1.88	L_OUT_EVERY_FORCES	99
6.6.1.89	L_OUTLET_ON	99
6.6.1.90	L_OUTPUT_PRECISION	99
6.6.1.91	L_PERIODIC_BOUNDARIES	100
6.6.1.92	L_PHYSICAL_U	100
6.6.1.93	L_PI	100
6.6.1.94	L_PROBE_OUT_FREQ	100
6.6.1.95	L_RE	100
6.6.1.96	L_RESOLUTION	100
6.6.1.97	L_RESTART_OUT_FREQ	100
6.6.1.98	L_RHOIN	100
6.6.1.99	L_SOLID_FROM_FILE	100
6.6.1.100	L_START_BFL_X	100
6.6.1.101	L_START_BFL_Y	101
6.6.1.102	L_START_IBB_X	101
6.6.1.103	L_START_IBB_Y	101
6.6.1.104	L_START_OBJECT_X	101
6.6.1.105	L_START_OBJECT_Y	101
6.6.1.106	L_TIMESTEP	101
6.6.1.107	L_TOTAL_TIMESTEPS	101
6.6.1.108	L_UMAX	101
6.6.1.109	L_UREF	101
6.6.1.110	L_UX0	101
6.6.1.111	L_UY0	102
6.6.1.112	L_UZ0	102

6.6.1.113	L_UZ0	102
6.6.1.114	L_VTK_BODY_WRITE	102
6.6.1.115	L_WALL_THICKNESS_BACK	102
6.6.1.116	L_WALL_THICKNESS_BOTTOM	102
6.6.1.117	L_WALL_THICKNESS_FRONT	102
6.6.1.118	L_WALL_THICKNESS_TOP	102
6.6.1.119	LUMA_VERSION	102
6.6.2	Variable Documentation	102
6.6.2.1	cNumProbes	102
6.6.2.2	cProbeLimsX	103
6.6.2.3	cProbeLimsY	103
6.6.2.4	cProbeLimsZ	103
6.6.2.5	cRefEndX	103
6.6.2.6	cRefEndY	103
6.6.2.7	cRefEndZ	103
6.6.2.8	cRefStartX	103
6.6.2.9	cRefStartY	104
6.6.2.10	cRefStartZ	104
6.7	Enumerations.h File Reference	104
6.7.1	Enumeration Type Documentation	105
6.7.1.1	eBCType	105
6.7.1.2	eCartesianDirection	105
6.7.1.3	eCartMinMax	105
6.7.1.4	eEdgeMinMax	105
6.7.1.5	eHdf5SlabType	106
6.7.1.6	eIInfoType	106
6.7.1.7	eIOFlag	106
6.7.1.8	eLocationOnRank	106
6.7.1.9	eMinMax	107
6.7.1.10	eObjectType	107

6.7.1.11 eType	107
6.8 GridManager.cpp File Reference	107
6.9 GridManager.h File Reference	108
6.10 GridObj.cpp File Reference	108
6.11 GridObj.h File Reference	108
6.12 GridObj_init_grids.cpp File Reference	108
6.13 GridObj_ops_boundary.cpp File Reference	108
6.14 GridObj_ops_io.cpp File Reference	109
6.15 GridObj_ops_lbm.cpp File Reference	109
6.16 GridObj_ops_lbm_optimised.cpp File Reference	109
6.17 GridUnits.h File Reference	109
6.18 GridUtils.cpp File Reference	109
6.19 GridUtils.h File Reference	109
6.20 hdf5luma.h File Reference	110
6.20.1 Macro Definition Documentation	110
6.20.1.1 H5_BUILT_AS_DYNAMIC_LIB	110
6.20.1.2 HDF5_EXT_SZIP	110
6.20.1.3 HDF5_EXT_ZLIB	110
6.20.2 Function Documentation	110
6.20.2.1 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, GridObj *g, T *data, hid_t hdf_datatype, bool *TL_present, int TL_thickness, HDFstruct hdf_data) . .	110
6.21 HDFstruct.h File Reference	111
6.22 IBody.cpp File Reference	111
6.23 IBody.h File Reference	111
6.24 IInfo.cpp File Reference	112
6.25 IInfo.h File Reference	112
6.26 IMarker.cpp File Reference	112
6.27 IMarker.h File Reference	112
6.28 IVector.h File Reference	112
6.29 main_lbm.cpp File Reference	113

6.29.1	Function Documentation	113
6.29.1.1	main(int argc, char *argv[])	113
6.30	Marker.h File Reference	113
6.31	MarkerData.h File Reference	113
6.32	Mpi_buffer_pack.cpp File Reference	114
6.33	Mpi_buffer_size_recv.cpp File Reference	114
6.34	Mpi_buffer_size_send.cpp File Reference	114
6.35	Mpi_buffer_unpk.cpp File Reference	114
6.36	MpiManager.cpp File Reference	114
6.37	MpiManager.h File Reference	114
6.37.1	Macro Definition Documentation	115
6.37.1.1	range_i_left	115
6.37.1.2	range_i_right	115
6.37.1.3	range_j_down	115
6.37.1.4	range_j_up	115
6.37.1.5	range_k_back	115
6.37.1.6	range_k_front	115
6.38	ObjectManager.cpp File Reference	116
6.39	ObjectManager.h File Reference	116
6.40	ObjectManager_init_bflbody.cpp File Reference	116
6.41	ObjectManager_init_ibmbody.cpp File Reference	116
6.42	ObjectManager_ops_ibm.cpp File Reference	116
6.43	ObjectManager_ops_ibmflex.cpp File Reference	117
6.43.1	Macro Definition Documentation	117
6.43.1.1	SWAP	117
6.43.1.2	SWAP	117
6.43.1.3	TINY	117
6.44	ObjectManager_ops_io.cpp File Reference	117
6.45	PCpts.h File Reference	117
6.46	stdafx.cpp File Reference	118

6.46.1	Variable Documentation	118
6.46.1.1	c	118
6.46.1.2	c_opt	118
6.46.1.3	cs	119
6.46.1.4	w	119
6.47	stdafx.h File Reference	119
6.47.1	Macro Definition Documentation	120
6.47.1.1	DEPRECATED	120
6.47.1.2	L_DACTION_WRITE_OUT_FORCES	120
6.47.1.3	L_ERROR	120
6.47.1.4	L_INFO	120
6.47.1.5	L_IS_NAN	120
6.47.1.6	L_SMALL_NUMBER	120
6.47.1.7	LUMA_FAILED	120
6.47.1.8	SQ	120
6.47.2	Function Documentation	120
6.47.2.1	errorfcn(const std::string &msg, std::ofstream *logfile)	120
6.47.2.2	infofcn(const std::string &msg, std::ofstream *logfile)	121
6.47.3	Variable Documentation	121
6.47.3.1	c	121
6.47.3.2	c_opt	121
6.47.3.3	cs	121
6.47.3.4	w	121

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	50
MpiManager::buffer_struct	16
GridManager	17
GridObj	20
GridUnits	31
GridUtils	33
HDFstruct	48
IBInfo	55
MpiManager::layer_edges	61
Marker	62
BFLMarker	11
IBMarker	56
MarkerData	64
MpiManager	66
ObjectManager	73
PCpts	82
vector	
IVector< GenTyp >	59
IVector< double >	59
IVector< eType >	59

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
GridManager		
	Grid Manager class	17
GridObj		
	Grid class	20
GridUnits		
	GridUnits	31
GridUtils		
	Grid utility class	33
HDFstruct		
	Structure for storing halo information for HDF5	48
IBBody		
	Immersed boundary body	50
IBInfo		
	Structure for passing IB information between MPI processes	55
IBMarker		
	Immersed boundary marker	56
IVector< GenTyp >		
	Index-collapsing vector class	59
MpiManager::layer_edges		
	Structure containing absolute positions of the edges of halos	61
Marker		
	Generic marker class	62
MarkerData		
	Container class to hold marker information	64
MpiManager		
	MPI Manager class	66
ObjectManager		
	Object Manager class	73
PCpts		
	Class to hold point cloud data	82

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

BFLBody.cpp	85
BFLBody.h	85
BFLMarker.cpp	85
BFLMarker.h	85
Body.h	86
definitions.h	86
Enumerations.h	104
GridManager.cpp	107
GridManager.h	108
GridObj.cpp	108
GridObj.h	108
GridObj_init_grids.cpp	108
GridObj_ops_boundary.cpp	108
GridObj_ops_io.cpp	109
GridObj_ops_lbm.cpp	109
GridObj_ops_lbm_optimised.cpp	109
GridUnits.h	109
GridUtils.cpp	109
GridUtils.h	109
hdf5luma.h	110
HDFstruct.h	111
IBBody.cpp	111
IBBody.h	111
IBInfo.cpp	112
IBInfo.h	112
IBMarker.cpp	112
IBMarker.h	112
IVector.h	112
main_lbm.cpp	113
Marker.h	113
MarkerData.h	113
Mpi_buffer_pack.cpp	114
Mpi_buffer_size_recv.cpp	114
Mpi_buffer_size_send.cpp	114
Mpi_buffer_unpk.cpp	114

MpiManager.cpp	114
MpiManager.h	114
ObjectManager.cpp	116
ObjectManager.h	116
ObjectManager_init_bflbody.cpp	116
ObjectManager_init_ibmbody.cpp	116
ObjectManager_ops_ibm.cpp	116
ObjectManager_ops_ibmflex.cpp	117
ObjectManager_ops_io.cpp	117
PCpts.h	117
stdafx.cpp	118
stdafx.h	119

Chapter 5

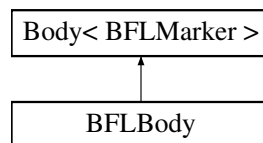
Class Documentation

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](#) ([GridObj](#) *g, [size_t](#) id, [PCpts](#) *_PCpts)
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 (GridObj * *g_hierarchy*, size_t *id*, PCpts * *_PCpts*)

Custom constructor to populate body from array of points.

Parameters

<i>g_hierarchy</i>	pointer to grid hierarchy
<i>id</i>	ID of body in array of bodies.
<i>_PCpts</i>	pointer to point cloud data

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:

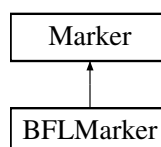
- [BFLBody.h](#)
- [BFLBody.cpp](#)

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, [GridObj](#) const *const body_owner)
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, \[GridObj\]\(#\) const *const body_owner \)](#)

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
<i>body_owner</i>	Grid on which primary support is to be found.

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.
- virtual [~Body](#) (void)
Default destructor.
- [Body](#) ([GridObj](#) *g, size_t id)
Custom constructor setting owning grid.
- [Body](#) ([GridObj](#) *g, size_t id, [PCpts](#) *_PCpts)
Custom constructor to call method to build from point cloud.

Protected Member Functions

- virtual 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 [passToVoxelFilter](#) (double x, double y, double z, int &curr_mark, std::vector< int > &counter)
Downsampling voxel-grid filter to take a point and add it to current body.
- void [buildFromCloud](#) ([PCpts](#) *_PCpts)
Method to build a body from point cloud data.

Protected Attributes

- double [spacing](#)
Reference spacing of the markers.
- 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 (i.e. last marker should link to first)
- [GridObj](#) * [_Owner](#)
Pointer to owning grid.
- size_t [id](#)
Unique ID of the body.

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) [virtual]`

Default destructor.

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

Custom constructor setting owning grid.

Parameters

<i>g</i>	pointer to grid which owns this body.
<i>id</i>	indicates unique number of body in array of bodies.

5.3.2.4 `template<typename MarkerType > Body< MarkerType >::Body (GridObj * g, size_t id, PCpts * _PCpts)`

Custom constructor to call method to build from point cloud.

Parameters

<i>g</i>	pointer to grid which owns this body.
<i>id</i>	indicates unique number of body in array of bodies.
<i>_PCpts</i>	pointer to point cloud data.

5.3.3 Member Function Documentation

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

Add marker to the body.

Parameters

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

5.3.3.2 `template<typename MarkerType > void Body< MarkerType >::buildFromCloud (PCpts * _PCpts)`
`[protected]`

Method to build a body from point cloud data.

Parameters

<i>_PCpts</i>	point cloud data from which to build body.
---------------	--

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

Retrieve marker data.

Return marker whose primary support data is nearest the 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.

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

Downsampling voxel-grid filter to take a point and add it to current body.

This method attempts to add a marker to body at the global location but obeys the rules of a voxel-grid filter to ensure markers are distributed such that their spacing roughly matches the background lattice. It is usually called in side a loop and requires a few extra pieces of information to be tracked throughout.

Parameters

<i>x</i>	desired global X-position of new marker.
<i>y</i>	desired global Y-position of new marker.
<i>z</i>	desired global Z-position of new marker.
<i>curr_mark</i>	is a reference to the ID of last marker added.
<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 (i.e. last marker should link to first)

5.3.4.3 `template<typename MarkerType> size_t Body< MarkerType >::id` `[protected]`

Unique ID of the body.

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

Array of markers which make up the body.

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

Reference spacing of the markers.

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_DIRS](#)]
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_DIRS]

Buffer sizes for each direction.

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

- [MpiManager.h](#)

5.5 GridManager Class Reference

Grid Manager class.

```
#include <GridManager.h>
```

Static Public Member Functions

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

Public Attributes

- [GridObj](#) * [Grids](#)
Pointer to grid hierarchy.

Protected Member Functions

- void [setLocalCoarseSize](#) (const std::vector< int > &size_vector)
Method to set the local coarse size.
- void [createWritableDataStore](#) (HDFstruct *&datastruct)
Method to create a blank store which holds the writable region information for a given grid.
- bool [createWritableDataStore](#) ([GridObj](#) const *const targetGrid)
Method to create and populate a writable data info store.

Protected Attributes

- int [global_size](#) [3][[L_NUM_LEVELS](#) * [L_NUM_REGIONS](#) + 1]
Overall size of each grid (excluding halo of course).
- double [global_edges](#) [6][[L_NUM_LEVELS](#) * [L_NUM_REGIONS](#) + 1]
Absolute position of grid edges (excluding halo of course).
- bool [subgrid_tlayer_key](#) [6][[L_NUM_LEVELS](#) * [L_NUM_REGIONS](#)]
Boolean flag array to indicate the presence of a TL on sub-grid edges.
- std::vector< int > [local_size](#)
Dimensions of coarsest lattice represented on this rank (includes halo if using MPI).
- std::vector< [HDFstruct](#) > [p_data](#)
Vector of structures containing writable region descriptors for block writing (HDF5)

Friends

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

5.5.1 Detailed Description

Grid Manager class.

Class to manage all information relating to GridObjs in the application. This singleton class may be accessed to supply information about local grids.

5.5.2 Member Function Documentation

5.5.2.1 void GridManager::createWritableDataStore ([HDFstruct](#) *& *datastruct*) [protected]

Method to create a blank store which holds the writable region information for a given grid.

Parameters

out	<i>datastruct</i>	reference to pointer to newly created data.
-----	-------------------	---

5.5.2.2 bool GridManager::createWritableDataStore ([GridObj](#) const *&const *targetGrid*) [protected]

Method to create and populate a writable data info store.

Parameters

<i>targetGrid</i>	constant pointer to constant grid to be used to populate information.
-------------------	---

Returns

boolean indicator as to whether writable data on this grid.

5.5.2.3 void GridManager::destroyInstance () [static]

Instance destroyer.

5.5.2.4 GridManager * GridManager::getInstance () [static]

Instance creator.

5.5.2.5 void GridManager::setLocalCoarseSize (const std::vector< int > & size_vector) [protected]

Method to set the local coarse size.

Accessible to the MPI manager so can be set remotely in parallel builds.

Parameters

<i>size_vector</i>	vector of sizes to assign.
--------------------	----------------------------

5.5.3 Friends And Related Function Documentation**5.5.3.1 friend class GridObj [friend]****5.5.3.2 friend class GridUtils [friend]****5.5.3.3 friend class MpiManager [friend]****5.5.4 Member Data Documentation****5.5.4.1 double GridManager::global_edges[6][L_NUM_LEVELS * L_NUM_REGIONS+1] [protected]**

Absolute position of grid edges (excluding halo of course).

Since L0 can only be region = 0 this array should be accessed as [level + region_number * L_NUM_LEVELS] in a loop where level cannot be 0. To retrieve L0 info, simply access [0]. The first index should be accessed using the eCartesianMinMax enumeration.

5.5.4.2 int GridManager::global_size[3][L_NUM_LEVELS * L_NUM_REGIONS+1] [protected]

Overall size of each grid (excluding halo of course).

Since L0 can only be region = 0 this array should be accessed as [level + region_number * L_NUM_LEVELS] in a loop where level cannot be 0. To retrieve L0 info, simply access [0]. The first index can be accessed using the eCartesianDirection enumeration.

5.5.4.3 GridObj* GridManager::Grids

Pointer to grid hierarchy.

5.5.4.4 std::vector<int> GridManager::local_size [protected]

Dimensions of coarsest lattice represented on this rank (includes halo if using MPI).

5.5.4.5 std::vector<HDFstruct> GridManager::p_data [protected]

Vector of structures containing writable region descriptors for block writing (HDF5)

5.5.4.6 bool GridManager::subgrid_tlayer_key[6][L_NUM_LEVELS * L_NUM_REGIONS] [protected]

Boolean flag array to indicate the presence of a TL on sub-grid edges.

It is not a given that a sub-grid has a TL on every edge of the grid. Specifically if we have a sub-grid which is periodic (or in future, which merges with another sub-grid?). The HDF5 writer needs to know whether to exclude sites to account for TL or not so we store information here from the sub-grid initialisation. The first index should be accessed using the enumerator eCartesianMinMax. If no sub-grids present then adopts a default 6x1 size to avoid a compilation error.

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

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

5.6 GridObj Class Reference

Grid class.

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

Public Member Functions

- [GridObj \(\)](#)
Default Constructor.
- [GridObj \(int level\)](#)
Serial build constructor for top level grid.
- [GridObj \(int RegionNumber, GridObj &pGrid\)](#)
Constructor for a sub-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](#) ()
- Method to initialise all L0 lattice quantities.*

 - void [LBM_initSubGrid](#) (GridObj &pGrid)
- Method to initialise all sub-grid quantities.*

 - void [LBM_initGridToGridMappings](#) (GridObj &pGrid)
- Method to initialise the mapping parameters between this grid and the supplied parent.*

 - void [LBM_initPositionVector](#) (double start_pos, double end_pos, [eCartesianDirection](#) dir)
- Method to initialise the position vector on the grid between the start and end positions supplied.*

 - 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_kbcCollide](#) (int i, int j, int k, [IVector](#)< double > &f_new)
- KBC collision operator.*

 - void [LBM_macro](#) (int i, int j, int k)
- Site-specific macroscopic update.*

 - void [LBM_resetForces](#) ()
- Method to reset body forces.*

 - **DEPRECATED** void [bc_applyBounceBack](#) (int label, int i, int j, int k)
- Method to apply half-way bounce-back.*

 - **DEPRECATED** void [bc_applySpecReflect](#) (int label, int i, int j, int k)
- Method to apply half-way specular reflection.*

 - **DEPRECATED** void [bc_applyRegularised](#) (int label, int i, int j, int k)
- Method to apply regularised velocity inlet.*

 - **DEPRECATED** void [bc_applyExtrapolation](#) (int label, int i, int j, int k)
- Method to apply extrapolation outlet.*

 - **DEPRECATED** void [bc_applyBfl](#) (int i, int j, int k)
- Method to apply BFL bounce-back.*

 - **DEPRECATED** void [bc_applyNrbc](#) (int i, int j, int k)
- Method to apply NRBC.*

 - 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_fgaout](#) ()
- .fga file writer.*

 - void [io_restart](#) ([eIOFlag](#) 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.*

 - void [LBM_multi_opt](#) (int subcycle=0)
- Optimised LBM multi-grid kernel.*

Public Attributes

- `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.
- `double dh`
Physical lattice spacing (same for x, y and z)
- `int region_number`
Region number.
- `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.
- `double XOrigin`
Position of grid left edge.
- `double YOrigin`
Position of grid bottom edge.
- `double ZOrigin`
Position of grid front edge.

Friends

- class `MpiManager`
- class `ObjectManager`
- class `GridUtils`

5.6.1 Detailed Description

Grid class.

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

5.6.2 Constructor & Destructor Documentation

5.6.2.1 GridObj::GridObj (void)

Default Constructor.

5.6.2.2 GridObj::GridObj (int *level*)

Serial build 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 as top level grid.
--------------	--

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

Constructor for a sub-grid.

This is not called directly but by the addSubGrid() method which first performs a check to see if a sub-grid is required.

Parameters

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

5.6.2.4 GridObj::~GridObj (void)

Default Destructor.

5.6.3 Member Function Documentation

5.6.3.1 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.

5.6.3.2 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.

5.6.3.3 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.
<i>i</i>	current site i-index.
<i>j</i>	current site j-index.
<i>k</i>	current site k-index.

5.6.3.4 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.

5.6.3.5 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.

5.6.3.6 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.

5.6.3.7 void GridObj::io_fgaout ()

.fga file writer.

Writes the components of the macroscopic velocity of the grid at time t and call recursively for any sub-grid. Writes the data of each subgrid in a different .fga file. .fga is the ASCII file format used by Unreal Engine 4 to read the data that populates a VectorField object. It doesn't do anything if the model is not 2D or 3D. Since .fga files can only store 3D data

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

HDF5 writer.

Useful grid quantities written out as scalar arrays. Creates one *.h5 file per grid and data is grouped into timesteps within each file. Should be used with the merge tool at post-processing to conver to structured VTK output readable in paraview.

Parameters

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

5.6.3.9 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.

5.6.3.10 void GridObj::io_probeOutput ()

Probe writer.

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

5.6.3.11 void GridObj::io_restart (eIOFlag *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 or read
----------------	--

5.6.3.12 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.
-------------------	--

5.6.3.13 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.
---------------------	---

5.6.3.14 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).

5.6.3.15 void GridObj::LBM_initBoundLab ()

Method to initialise wall and object labels on L0.

The virtual wind tunnel definitions are implemented by this method.

5.6.3.16 void GridObj::LBM_initGrid ()

Method to initialise all L0 lattice quantities.

5.6.3.17 void GridObj::LBM_initGridToGridMappings (GridObj & pGrid)

Method to initialise the mapping parameters between this grid and the supplied parent.

The mappings computed by this method are local ijk references to allow correct coupling during multi-grid operations.

Parameters

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

5.6.3.18 void GridObj::LBM_initPositionVector (double start_pos, double end_pos, eCartesianDirection dir)

Method to initialise the position vector on the grid between the start and end positions supplied.

This method can be used for either serial or parallel initialisation as the halo and any wrap around is automatically taken into consideration. As such, the start position can be after the end position and the resulting vector will wrap at the correct point.

Parameters

<i>start_pos</i>	position of the first voxel centre in the vector.
<i>end_pos</i>	position of the last voxel centre in the vector.
<i>dir</i>	direction of the vector.

5.6.3.19 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.
--------------	---------------------------

5.6.3.20 void GridObj::LBM_initRho ()

Method to initialise the lattice density.

5.6.3.21 void GridObj::LBM_initSolidLab ()

Method to initialise label-based solids.

5.6.3.22 void GridObj::LBM_initSubGrid (GridObj & pGrid)

Method to initialise all sub-grid quantities.

Parameters

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

5.6.3.23 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.

5.6.3.24 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.

5.6.3.25 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.

5.6.3.26 void GridObj::LBM_multi_opt (int *subcycle* = 0)

Optimised LBM multi-grid kernel.

This kernel compresses the old kernel into a single loop in order to make it more efficient. Capabilities are current limited with this kernel with incompatible options giving unpredictable results. Use with caution.

Parameters

<i>subcycle</i>	sub-cycle to be performed if called from a subgrid.
-----------------	---

5.6.3.27 void GridObj::LBM_resetForces ()

Method to reset body forces.

Resets both Cartesian and Lattice force vectors to zero.

5.6.4 Friends And Related Function Documentation

5.6.4.1 friend class GridUtils [friend]

5.6.4.2 friend class MpiManager [friend]

5.6.4.3 friend class ObjectManager [friend]

5.6.5 Member Data Documentation

5.6.5.1 double GridObj::dh

Physical lattice spacing (same for x, y and z)

5.6.5.2 double GridObj::dt

Physical time step size.

5.6.5.3 int GridObj::K_lim

Local size of grid in Z-direction.

5.6.5.4 `IVector<eType> GridObj::LatTyp`

Flattened 3D array of site labels.

5.6.5.5 `int GridObj::level`

Level in embedded grid hierarchy.

5.6.5.6 `int GridObj::M_lim`

Local size of grid in Y-direction.

5.6.5.7 `int GridObj::N_lim`

Local size of grid in X-direction.

5.6.5.8 `double GridObj::nu`

Kinematic viscosity (in lattice units)

5.6.5.9 `double GridObj::omega`

Relaxation frequency.

5.6.5.10 `int GridObj::region_number`

Region number.

5.6.5.11 `int GridObj::t`

Number of completed iterations on this level.

5.6.5.12 `double GridObj::timeav_mpi_overhead`

Time-averaged time of MPI communication.

5.6.5.13 `double GridObj::timeav_timestep`

Time-averaged time of a timestep.

5.6.5.14 double GridObj::XOrigin

Position of grid left edge.

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

Vector of global X positions of each site.

5.6.5.16 double GridObj::YOrigin

Position of grid bottom edge.

5.6.5.17 std::vector<double> GridObj::YPos

Vector of global Y positions of each site.

5.6.5.18 double GridObj::ZOrigin

Position of grid front edge.

5.6.5.19 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](#)
- [GridObj_ops_lbm_optimised.cpp](#)

5.7 GridUnits Class Reference

[GridUnits](#).

```
#include <GridUnits.h>
```

Public Member Functions

- [GridUnits](#) ()
- [~GridUnits](#) ()

Static Public Member Functions

- `template<typename T >`
`static T m2cm (const T meters)`
Convert from m to cm.
- `template<typename T >`
`static T ulat2uphys (T ulat, GridObj *currentGrid)`
Velocity in lattice units to velocity in physical units.

5.7.1 Detailed Description

[GridUnits](#).

This class contains static methods for unit conversion (the only ones implemented are from m to cm and velocity from lattice units to m/s)

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `GridUnits::GridUnits ()` `[inline]`

5.7.2.2 `GridUnits::~~GridUnits ()` `[inline]`

5.7.3 Member Function Documentation

5.7.3.1 `template<typename T > static T GridUnits::m2cm (const T meters)` `[inline]`, `[static]`

Convert from m to cm.

5.7.3.2 `template<typename T > static T GridUnits::ulat2uphys (T ulat, GridObj * currentGrid)` `[inline]`, `[static]`

Velocity in lattice units to velocity in physical units.

Converts velocity component from lattice units to m/s. It uses the `L_PHYSICAL_U` introduced by the user, `dh` and `dt`. You can introduce any `L_PHYSICAL_U` you want, but the reference lenght (usualy the width of the domain) , the Re number and the LBM parameters will remain the same. So you will be implicitly changing the physical viscosity of your fluid when you change `L_PHYSICAL_U`

Parameters

<i>ulat</i>	Lattice velocity.
<i>currentGrid</i>	Pointer to the current grid.

Returns

physical velocity

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

- [GridUnits.h](#)

5.8 GridUtils Class Reference

Grid utility class.

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

Static Public Member Functions

- static void [createOutputDirectory](#) (std::string [path_str](#))
Create output directory.
- static std::vector< int > [onespace](#) (int min, int max)
Creates a linearly-spaced vector of integers.
- static std::vector< double > [linspace](#) (double min, double max, int n)
Creates a linearly-spaced vector of values.
- static double [vecnorm](#) (double vec[[L_DIMS](#)])
Computes the L2 norm using the vector supplied.
- static double [vecnorm](#) (double val1, double val2)
Computes the L2 norm using the vector components supplied.
- static double [vecnorm](#) (double val1, double val2, double val3)
Computes the L2 norm using the vector components supplied.
- static double [vecnorm](#) (std::vector< double > vec)
Computes the L2 norm using the vector supplied.
- static std::vector< int > [getFineIndices](#) (int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)
Gets the indices of the fine site given the coarse site.
- static std::vector< int > [getCoarseIndices](#) (int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)
Gets the indices of the coarse site given the fine site.
- static double [dotprod](#) (std::vector< double > vec1, std::vector< double > vec2)
Computes the scalar product of two vectors.
- static std::vector< double > [subtract](#) (std::vector< double > a, std::vector< double > b)
Subtracts two vectors.
- static std::vector< double > [add](#) (std::vector< double > a, std::vector< double > b)
Adds two vectors.
- static std::vector< double > [vecmultiply](#) (double scalar, std::vector< double > vec)
Multiplies a scalar by a vector.
- static std::vector< double > [crossprod](#) (std::vector< double > vec1, std::vector< double > vec2)
Computes vector product.
- static std::vector< double > [matrix_multiply](#) (const std::vector< std::vector< double > > &A, const std::vector< double > &x)
Multiplies matrix A by vector x.
- static int [getOpposite](#) (int direction)
Gets the opposite lattice direction to the one supplied.
- static void [getGrid](#) ([GridObj](#) *const Grids, int level, int region, [GridObj](#) *&ptr)
Get a pointer to a given grid in the hierarchy.
- static bool [isOverlapPeriodic](#) (int i, int j, int k, [GridObj](#) const &pGrid)
Finds out whether halo containng i,j,k links to neighbour rank periodically.

- static bool [isOnThisRank](#) (double x, double y, double z, [eLocationOnRank](#) *loc=nullptr, [GridObj](#) const *const grid=nullptr, std::vector< int > *pos=nullptr)
Finds out whether site with supplied position is on the current rank.
- static bool [isOnThisRank](#) (double xyz, [eCartesianDirection](#) dir, [eLocationOnRank](#) *loc=nullptr, [GridObj](#) const *const grid=nullptr, int *pos=nullptr)
Finds out whether the supplied position can be found on the current rank.
- static bool [intersectsRefinedRegion](#) ([GridObj](#) const &pGrid, int RegNum)
Finds out whether all or part of specified refined region intersects with the space occupied by the grid provided.
- static bool [isOnSenderLayer](#) (double pos_x, double pos_y, double pos_z)
Check whether site is on an inner (sender) halo.
- static bool [isOnRecvLayer](#) (double pos_x, double pos_y, double pos_z)
Check whether site is on an outer (receiver) halo.
- static bool [isOnSenderLayer](#) (double site_position, [eCartMinMax](#) edge)
Check whether site is on an inner (sender) halo.
- static bool [isOnRecvLayer](#) (double site_position, [eCartMinMax](#) edge)
Check whether site is on an outer (receiver) halo.
- static int [getMpiDirection](#) (int offset_vector[])
Get direction in MPI topology from unit vector.
- static int [safeGetRank](#) ()
Safe method to get the rank number.
- static bool [isOffGrid](#) (int i, int j, int k, [GridObj](#) const *const g)
Tests whether a site is on a given grid.
- static void [getEnclosingVoxel](#) (double x, double y, double z, [GridObj](#) const *const g, std::vector< int > *ijk)
Get local voxel indices on grid in which provided position lies.
- static void [getEnclosingVoxel](#) (double x, [GridObj](#) const *const g, [eCartesianDirection](#) dir, int *ijk)
Get local voxel indices on grid in which provided position lies.
- static bool [isOnTransitionLayer](#) (double pos_x, double pos_y, double pos_z, [GridObj](#) const *const grid)
Check whether site is on a TL.
- static bool [isOnTransitionLayer](#) (double position, [eCartMinMax](#) edge, [GridObj](#) const *const grid)
Check whether site is on a specific TL (to upper).
- 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.

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_NUM_VELS](#)]
Array with hardcoded direction numbering for specular reflection.

5.8.1 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.

5.8.2 Member Function Documentation

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

Adds two vectors.

Parameters

<i>a</i>	a vector.
<i>b</i>	a second vector.

Returns

vector which is $a + b$.

5.8.2.2 `void GridUtils::createOutputDirectory (std::string path_str)` `[static]`

Create output directory.

Compatible with both Windows and Linux. Filename and path passed as a single string. Returns nothing at the moment.

Parameters

<i>path_str</i>	full path and filename as string.
-----------------	-----------------------------------

Returns

indicator of status of action.

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

Computes vector product.

Parameters

<i>a</i>	a vector.
<i>b</i>	a second vector.

Returns

a vector which is the cross product of *a* and *b*.

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

Computes the scalar product of two vectors.

Parameters

<i>vec1</i>	a vector.
<i>vec2</i>	a second vector.

Returns

the dot product of the two vectors.

5.8.2.5 `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

5.8.2.6 `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

n	factorial
-----	-----------

Returns

NumType n factorial

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

Gets the indices of the coarse site given the fine site.

Maps the indices of a fine grid site to a corresponding coarse site on the level above.

Parameters

<i>fine_i</i>	local i-index of fine site to be mapped.
<i>x_start</i>	local x-index of start of refined region on the grid above.
<i>fine_j</i>	local j-index of fine site to be mapped.
<i>y_start</i>	local y-index of start of refined region on the grid above.
<i>fine_k</i>	local k-index of fine site to be mapped.
<i>z_start</i>	local z-index of start of refined region on the grid above.

Returns

local indices of the coarse grid site.

5.8.2.8 `void GridUtils::getEnclosingVoxel (double x, double y, double z, GridObj const *const g, std::vector< int > * ijk)`
`[static]`

Get local voxel indices on grid in which provided position lies.

Wrapper for the overload which concentrates all check into a vector.

Parameters

<i>x</i>	x-position.
<i>y</i>	y-position.
<i>z</i>	z-position.
<i>g</i>	lattice on which to look for enclosing voxel.
<i>ijk</i>	pointer to vector where indices are to be placed.

5.8.2.9 `void GridUtils::getEnclosingVoxel (double xyz, GridObj const *const g, eCartesianDirection dir, int * ijk)`
`[static]`

Get local voxel indices on grid in which provided position lies.

Will return the 1D voxel index of the voxel on the lattice provided within which point with position (xyz) lies. This is done by rounding the position to obtain how many voxels in from the grid core edge it is, then accounting for whether the grid starts on another rank, in the halo, or further into the grid by offsetting the original index by this amount. This approach saves expensive searches of the position vectors on each grid. This method can be used as a position -> voxel converter. The index may be off grid so it is advisable to call `isOnThisRank` instead.

Parameters

<i>xyz</i>	x, y or z-position.
<i>g</i>	lattice on which to look for enclosing voxel.
<i>dir</i>	1D direction.
<i>ijk</i>	pointer to local index storage location.

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

Gets the indices of the fine site given the coarse site.

Maps the indices of a coarse grid site to a corresponding fine site on the level below.

Parameters

<i>coarse_i</i>	local i-index of coarse site to be mapped.
<i>x_start</i>	local x-index of start of refined region.
<i>coarse_j</i>	local j-index of coarse site to be mapped.
<i>y_start</i>	local y-index of start of refined region.
<i>coarse_k</i>	local k-index of coarse site to be mapped.
<i>z_start</i>	local z-index of start of refined region.

Returns

local indices of the fine grid site.

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

Get a pointer to a given grid in the hierarchy.

Takes a NULL pointer by reference and updates it when matching grid is found in hierarchy on this rank. If grid not found, pointer is returned without change and stays NULL. Can be used to test for the existence of a grid on a rank by passing in a NULL pointer and checking if a NULL pointer is returned.

Parameters

	<i>Grids</i>	constant pointer to the grid at which to start searching.
	<i>level</i>	level desired.
	<i>region</i>	region desired.
out	<i>ptr</i>	reference to pointer where address of grid matching in hierarchy will be assigned.

5.8.2.12 `int GridUtils::getMpiDirection (int offset_vector[]) [static]`

Get direction in MPI topology from unit vector.

Parameters

<i>offset_vector</i>	unit vector pointing away from current rank.
----------------------	--

Returns

MPI direction.

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

Gets the opposite lattice direction to the one supplied.

This is model independent as long as the model directions are specified such that the opposite direction is either one vector on or one vector back in the listing depending on whether the direction supplied is even or odd.

Parameters

<i>direction</i>	direction to be reversed.
------------------	---------------------------

Returns

opposite direction in lattice model.

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

Finds out whether all or part of specified refined region intersects with the space occupied by the grid provided.

Principal use is for sub-grid initialisation to determine whether a sub-grid needs adding or not. This decision is made based on whether any part of the grid is covered by the discrete voxels of existing grids on the rank.

Parameters

<i>pGrid</i>	parent grid at appropriate level.
<i>RegNum</i>	region number desired.

Returns

boolean answer.

5.8.2.15 `bool GridUtils::isOffGrid (int i, int j, int k, GridObj const *const g)` `[static]`

Tests whether a site is on a given grid.

Parameters

<i>i</i>	local i-index.
<i>j</i>	local j-index.
<i>k</i>	local k-index.
<i>g</i>	grid on which to check.

Returns

boolean answer.

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

Check whether site is on an outer (receiver) halo.

Wrapper which checks every halo region of the rank for intersection with supplied site position.

Parameters

<i>pos</i> ↔ _x	x-position of site.
<i>pos</i> ↔ _y	y-position of site.
<i>pos</i> ↔ _z	z-position of site.

Returns

boolean answer.

5.8.2.17 `bool GridUtils::isOnRecvLayer (double site_position, eCartMinMax edge)` `[static]`

Check whether site is on an outer (receiver) halo.

Wrapper available which checks every halo. This method only checks the halo specified by the Cartesian direction and whether it is the left/bottom/front (minimum) or right/top/back (maximum) edge of the block.

Parameters

<i>site_position</i>	position of site.
<i>edge</i>	combination of cartesian direction and choice of edge.

Returns

boolean answer.

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

Check whether site is on an inner (sender) halo.

Wrapper which checks every halo region of the rank for intersection with supplied site position.

Parameters

<i>pos_x</i>	x-position of site.
<i>pos_y</i>	y-position of site.
<i>pos_z</i>	z-position of site.

Returns

boolean answer.

5.8.2.19 `bool GridUtils::isOnSenderLayer (double site_position, eCartMinMax edge) [static]`

Check whether site is on an inner (sender) halo.

Wrapper available which checks every halo. This method only checks the halo specified by the Cartesian direction and whether it is the left/bottom/front (minimum) or right/top/back (maximum) edge of the block.

Parameters

<i>site_position</i>	position of site.
<i>edge</i>	combination of cartesian direction and choice of edge.

Returns

boolean answer.

5.8.2.20 `bool GridUtils::isOnThisRank (double x, double y, double z, eLocationOnRank * loc = nullptr, GridObj const * grid = nullptr, std::vector< int > * pos = nullptr) [static]`

Finds out whether site with supplied position is on the current rank.

Will return true if the site is in the halo as well (send or recv). Location information provided to indicate where point is. Returns eNone enumeration if not request or if query is false. If a grid is supplied, will only return true if site is on the grid supplied. If you want to exclude the sites that belong to the halo you can call [isOnRecvLayer\(\)](#) or [isOnSenderLayer\(\)](#) on the same site.

Parameters

	<i>x</i>	x-position of site.
	<i>y</i>	y-position of site.
	<i>z</i>	z-position of site.
out	<i>pos</i>	pointer to the start of a vector in which local indices are returned.
	<i>grid</i>	grid being queried.
out	<i>loc</i>	description of the location of the point.

Returns

boolean answer.

5.8.2.21 `bool GridUtils::isOnThisRank (double xyz, eCartesianDirection dir, eLocationOnRank * loc = nullptr, GridObj const *const grid = nullptr, int * pos = nullptr) [static]`

Finds out whether the supplied position can be found on the current rank.

Direction-specific version of the overload.

Parameters

	<i>xyz</i>	position (x, y or z)
	<i>dir</i>	cartesian direction of interest (x, y or z).
out	<i>loc</i>	description of the location of the point.
	<i>grid</i>	grid being queried.
out	<i>pos</i>	the local index of the found site.

Returns

boolean answer.

5.8.2.22 `bool GridUtils::isOnTransitionLayer (double pos_x, double pos_y, double pos_z, GridObj const *const grid) [static]`

Check whether site is on a TL.

Wrapper which checks every possible TL location on the grid supplied.

Parameters

<i>pos</i> ↔ <i>_x</i>	x-position of site.
<i>pos</i> ↔ <i>_y</i>	y-position of site.
<i>pos</i> ↔ <i>_z</i>	z-position of site.
<i>grid</i>	given grid on which to check.

Returns

boolean answer.

5.8.2.23 `bool GridUtils::isOnTransitionLayer (double position, eCartMinMax edge, GridObj const *const grid)`
`[static]`

Check whether site is on a specific TL (to upper).

Wrapper available which checks every TL. This method only checks the TL specified by the Cartesian direction and whether it is the left/bottom/front (minimum) or right/top/back (maximum) edge of the supplied grid.

Parameters

<i>position</i>	position of point.
<i>edge</i>	combination of cartesian direction and choice of edge.
<i>grid</i>	given grid on which to check.

Returns

boolean answer.

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

Finds out whether halo containing *i,j,k* links to neighbour rank periodically.

Checks the receiver layer containing local site *i,j,k* and determines from the MPI topology information whether this layer couples to an adjacent or periodic neighbour rank. I.e. if the neighbour is physically next to the rank or whether it is actually at the other side of the domain.

Parameters

<i>i</i>	local i-index of recv layer site being queried.
<i>j</i>	local j-index of recv layer site being queried.
<i>k</i>	local k-index of recv layer site being queried.
<i>g</i>	grid on which point being queried resides.

Returns

boolean answer.

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

Creates a linearly-spaced vector of values.

Parameters

<i>min</i>	starting value of output vector.
<i>max</i>	ending point of output vector.
<i>n</i>	number of values in output vector.

Returns

a vector with n uniformly spaced values between min and max.

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

Multiplies matrix A by vector x.

Parameters

<i>A</i>	a matrix represented as a vector or vectors.
<i>x</i>	a vector.

Returns

a vector which is $A * x$.

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

Creates a linearly-spaced vector of integers.

Parameters

<i>min</i>	starting value of output vector.
<i>max</i>	ending point of output vector.

Returns

a vector with uniformly spaced integer values between min and max.

5.8.2.28 `int GridUtils::safeGetRank () [static]`

Safe method to get the rank number.

This is a serial/parallel agnostic method to get the rank number. This is necessary as often we just want to access the rank number for logging purposes and don't want to have to wrap every call to avoid attempts to use the MPI manager in non-MPI code.

Returns

integer specifying the rank number. Zero if using serial code.

5.8.2.29 `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.

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

Subtracts two vectors.

Parameters

<i>a</i>	a vector.
<i>b</i>	a second vector.

Returns

a vector which is $a - b$.

5.8.2.31 `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

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

Multiplies a scalar by a vector.

Parameters

<i>scalar</i>	a scalar double.
<i>vec</i>	a vector double.

Returns

a vector which is a scalar multiplied by a vector.

5.8.2.33 `double GridUtils::vecnorm (double vec[L_DIMS]) [static]`

Computes the L2 norm using the vector supplied.

Parameters

<i>vec</i>	old-style C array representing a vector with the same number of number of components as the problem dimension.
------------	--

Returns

the L2 norm.

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

Computes the L2 norm using the vector components supplied.

Parameters

<i>val1</i>	first vector component.
<i>val2</i>	second vector component.

Returns

the L2 norm.

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

Computes the L2 norm using the vector components supplied.

Parameters

<i>val1</i>	first vector component.
<i>val2</i>	second vector component.
<i>val3</i>	third vector component.

Returns

the L2 norm.

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

Computes the L2 norm using the vector supplied.

Parameters

<i>vec</i>	C++ std::vector.
------------	------------------

Returns

the L2 norm.

5.8.2.37 `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

5.8.2.38 `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

5.8.3 Member Data Documentation

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

Initial value:

```
=
{
    {1, 0, 2, 3, 7, 6, 5, 4, 8},
    {1, 0, 2, 3, 4, 6, 5, 4, 8},
    {0, 1, 3, 2, 6, 7, 4, 5, 8},
    {0, 1, 3, 2, 6, 7, 4, 5, 8}
}
```

Array with hardcoded direction numbering for specular reflection.

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

Handle to output file.

5.8.3.3 `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_lbm.cpp](#)

5.9 HDFstruct Struct Reference

Structure for storing halo information for HDF5.

```
#include <HDFstruct.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 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.

5.9.1 Detailed Description

Structure for storing halo information for HDF5.

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

5.9.2 Member Data Documentation

5.9.2.1 int HDFstruct::i_end

Ending i-index for writable region.

5.9.2.2 int HDFstruct::i_start

Starting i-index for writable region.

5.9.2.3 int HDFstruct::j_end

Ending j-index for writable region.

5.9.2.4 int HDFstruct::j_start

Starting j-index for writable region.

5.9.2.5 int HDFstruct::k_end

Ending k-index for writable region.

5.9.2.6 int HDFstruct::k_start

Starting k-index for writable region.

5.9.2.7 int HDFstruct::level

Grid level to which these data correspond.

5.9.2.8 int HDFstruct::region

Region number to which these data correspond.

5.9.2.9 unsigned int HDFstruct::writable_data_count = 0

Writable data count.

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

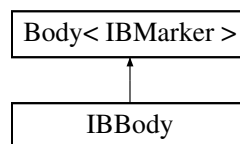
- [HDFstruct.h](#)

5.10 IBBody Class Reference

Immersed boundary body.

```
#include <IBBody.h>
```

Inheritance diagram for IBBody:



Public Member Functions

- [IBBody](#) (void)
Constructor which sets group ID to zero by default.
- [~IBBody](#) (void)
Default destructor.
- [IBBody](#) (GridObj *g, size_t id)
Constructor which assigns the owner grid.
- [IBBody](#) (GridObj *g, size_t id, PCpts * _PCpts)
Constructor to build a body in place using a point cloud..
- void [makeBody](#) (double radius, std::vector< double > centre, bool [isFlexible](#), bool [isMovable](#), 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 [isFlexible](#), bool [isMovable](#), 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 [isFlexible](#), bool [isMovable](#), int group)
Method to seed markers for a flexible filament.
- double [makeBody](#) (std::vector< double > width_length, double angle, std::vector< double > centre, bool [isFlexible](#), bool [isMovable](#), int group, bool plate)
Method to seed markers for a 3D plate inclined from the XZ plane.
- void [addMarker](#) (double x, double y, double z, bool [isFlexible](#))
Method to add an IB marker to the body.

Protected Attributes

- bool `isFlexible`
Flag to indicate flexibility: false == rigid body; true == flexible filament.
- bool `isMovable`
Flag to indicate if body is movable or not.
- int `groupID`
ID of IBody 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`
- class `IBInfo`

Additional Inherited Members

5.10.1 Detailed Description

Immersed boundary body.

5.10.2 Constructor & Destructor Documentation

5.10.2.1 IBody::IBody (void)

Constructor which sets group ID to zero by default.

5.10.2.2 IBody::~~IBody (void)

Default destructor.

5.10.2.3 IBody::IBody (GridObj * g, size_t id)

Constructor which assigns the owner grid.

Also sets the group ID to zero.

Parameters

<i>g</i>	pointer to owner grid
<i>id</i>	ID of body in array of bodies.

5.10.2.4 IBody::IBody (GridObj * *g*, size_t *id*, PCpts * *_PCpts*)

Constructor to build a body in place using a point cloud,.

isFlexible and isMovable properties taken from definitions.

Parameters

<i>g</i>	pointer to owner grid
<i>id</i>	ID of body in array of bodies.
<i>_PCpts</i>	pointer to point cloud data.

5.10.3 Member Function Documentation

5.10.3.1 void IBody::addMarker (double *x*, double *y*, double *z*, bool *isFlexible*)

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

5.10.3.2 void IBody::makeBody (double *radius*, std::vector< double > *centre*, bool *isFlexible*, bool *isMovable*, 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>isFlexible</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>isMovable</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.

5.10.3.3 void IBBody::makeBody (std::vector< double > *width_length_depth*, std::vector< double > *angles*, std::vector< double > *centre*, bool *isFlexible*, bool *isMovable*, 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>isFlexible</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>isMovable</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.

5.10.3.4 void IBBody::makeBody (int *nummarkers*, std::vector< double > *start_point*, double *fil_length*, std::vector< double > *angles*, std::vector< int > *BCs*, bool *isFlexible*, bool *isMovable*, 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>isFlexible</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>isMovable</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.

5.10.3.5 double IBBody::makeBody (std::vector< double > *width_length*, double *angle*, std::vector< double > *centre*, bool *isFlexible*, bool *isMovable*, 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>isFlexible</i>	flag to indicate whether body is flexible and requires a structural calculation.
<i>isMovable</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.

5.10.4 Friends And Related Function Documentation

5.10.4.1 `friend class IBInfo` `[friend]`

5.10.4.2 `friend class ObjectManager` `[friend]`

5.10.5 Member Data Documentation

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

BCs type flags (flexible bodies)

5.10.5.2 `double IBBody::delta_rho` `[protected]`

Difference in density between fluid and solid in lattice units.

5.10.5.3 `double IBBody::flexural_rigidity` `[protected]`

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

5.10.5.4 `int IBBody::groupID` `[protected]`

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

5.10.5.5 `bool IBBody::isFlexible` `[protected]`

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

5.10.5.6 `bool IBBody::isMovable` `[protected]`

Flag to indicate if body is movable or not.

5.10.5.7 `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](#)

5.11 IInfo Class Reference

Structure for passing IB information between MPI processes.

```
#include <IInfo.h>
```

Public Member Functions

- [IInfo](#) ()
- [IInfo](#) (IBBody *iBody, eIInfoType type)
Custom constructor for different types of message containers.
- int [mapToMpiStruct](#) (eIInfoType type)
Maps a version of the [IInfo](#) structure to an MPI_Struct datatype.

5.11.1 Detailed Description

Structure for passing IB information between MPI processes.

This structure has a series of different constructors depending on what information should be passed.

5.11.2 Constructor & Destructor Documentation

5.11.2.1 IInfo::IInfo ()

5.11.2.2 IInfo::IInfo (IBBody * iBody, eIInfoType type)

Custom constructor for different types of message containers.

Parameters

<i>iBody</i>	pointer to the iBody being packed.
<i>type</i>	the type fo container to be created.

5.11.3 Member Function Documentation

5.11.3.1 int IInfo::mapToMpiStruct (eIInfoType type)

Maps a version of the [IInfo](#) structure to an MPI_Struct datatype.

Parameters

<i>type</i>	type of container you want to map.
-------------	------------------------------------

Returns

handle to the MPI struct data.

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

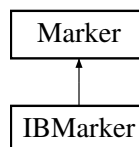
- [IBInfo.h](#)
- [IBInfo.cpp](#)

5.12 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, [GridObj](#) const *const body_owner, bool isFlexible=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 [isFlexible](#)
Indication as to whether marker is part of a structural or moving body calculation.
- 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](#)
- class [IBInfo](#)

Additional Inherited Members

5.12.1 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.

5.12.2 Constructor & Destructor Documentation

5.12.2.1 IBMarker::IBMarker (void) [inline]

Default constructor.

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

Default destructor.

5.12.2.3 IBMarker::IBMarker (double *xPos*, double *yPos*, double *zPos*, GridObj const *const *body_owner*, bool *isFlexible* = 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>body_owner</i>	Grid on which primary support point is to be found
<i>isFlexible</i>	flag to indicate whether marker is movable or not.

5.12.3 Friends And Related Function Documentation

5.12.3.1 friend class IBBody [friend]

5.12.3.2 friend class IBInfo [friend]

5.12.3.3 **friend class `ObjectManager`** `[friend]`

5.12.4 Member Data Documentation

5.12.4.1 **`std::vector<double> IBMarker::deltaval`** `[protected]`

Value of delta function for a given support node.

5.12.4.2 **`std::vector<double> IBMarker::desired_vel`** `[protected]`

Desired velocity at marker.

5.12.4.3 **`double IBMarker::dilation`** `[protected]`

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

5.12.4.4 **`double IBMarker::epsilon`** `[protected]`

Scaling parameter.

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

Fluid velocity interpolated from lattice nodes.

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

Restorative force vector on marker.

5.12.4.7 **`bool IBMarker::isFlexible`** `[protected]`

Indication as to whether marker is part of a structural or moving body calculation.

5.12.4.8 **`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)

5.12.4.9 **`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:

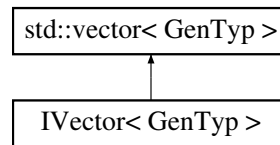
- [IBMarker.h](#)
- [IBMarker.cpp](#)

5.13 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.

5.13.1 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.

5.13.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

5.13.2.3 `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

5.13.3 Member Function Documentation

5.13.3.1 `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

5.13.3.2 `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

5.13.3.3 `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](#)

5.14 MpiManager::layer_edges Struct Reference

Structure containing absolute 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.

5.14.1 Detailed Description

Structure containing absolute 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 and similar for Y and Z. Access using the enumerator eEdgeMinMax.

5.14.2 Member Data Documentation

5.14.2.1 double MpiManager::layer_edges::X[4]

X limits.

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

Y limits.

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

Z limits.

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

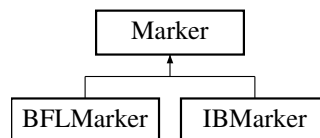
- [MpiManager.h](#)

5.15 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, [GridObj](#) const *const body_owner)
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.

5.15.1 Detailed Description

Generic marker class.

5.15.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

5.15.2.3 `Marker::Marker (double x, double y, double z, GridObj const *const body_owner)` `[inline]`

Custom constructor which locates marker.

In order to properly initialise during construction, a grid should be passed in on which primary support point can be found.

Parameters

<i>x</i>	X-position of marker
<i>y</i>	Y-position of marker
<i>z</i>	Z-position of marker
<i>body_owner</i>	Grid on which primary support point is to be found.

5.15.3 Member Data Documentation

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

Position vector of marker location in physical units.

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

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

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

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

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

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

5.15.3.5 `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](#)

5.16 MarkerData Class Reference

Container class to hold marker information.

```
#include <MarkerData.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

5.16.1 Detailed Description

Container class to hold marker information.

5.16.2 Constructor & Destructor Documentation

5.16.2.1 `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

5.16.2.2 MarkerData::MarkerData (void) [inline]

Default Constructor.

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

5.16.2.3 MarkerData::~~MarkerData (void) [inline]

Default destructor.

5.16.3 Member Data Documentation

5.16.3.1 int MarkerData::i

i-index of primary support site

5.16.3.2 int MarkerData::ID

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

5.16.3.3 int MarkerData::j

j-index of primary support site

5.16.3.4 int MarkerData::k

k-index of primary support site

5.16.3.5 double MarkerData::x

x-position of marker

5.16.3.6 double MarkerData::y

y-position of marker

5.16.3.7 double MarkerData::z

z-position of marker

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

- [MarkerData.h](#)

5.17 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 absolute positions of the edges of halos.

Public Member Functions

- void [mpi_init](#) ()
Initialisation routine.
- void [mpi_gridbuild](#) (GridManager *const grid_man)
Domain decomposition.
- int [mpi_buildCommunicators](#) (GridManager *const grid_man)
Define writable sub-grid communicators.
- void [mpi_updateLoadInfo](#) (GridManager *const grid_man)
Update the load balancing information stored in the [MpiManager](#).
- void [mpi_buffer_pack](#) (int dir, GridObj *const g)
Method to pack the communication buffer.
- void [mpi_buffer_unpack](#) (int dir, GridObj *const g)
Method to unpack the communication buffer.
- void [mpi_buffer_size](#) ()
Pre-calculation of the buffer sizes.
- void [mpi_buffer_size_send](#) (GridObj *const g)
Method to pre-compute the size of the sender layer buffer.
- void [mpi_buffer_size_rcv](#) (GridObj *const 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 [dimensions](#) [[L_DIMS](#)]
Size of MPI Cartesian topology.
- int [neighbour_rank](#) [[L_MPI_DIRS](#)]
Neighbour rank number for each direction in Cartesian topology.
- int [neighbour_coords](#) [[L_DIMS](#)][[L_MPI_DIRS](#)]
Coordinates in MPI topology of neighbour ranks.
- std::vector< int > [cRankSizeX](#)
Number of sites in X direction for each custom rank.
- std::vector< int > [cRankSizeY](#)
Number of sites in Y direction for each custom rank.
- std::vector< int > [cRankSizeZ](#)
Number of sites in Z direction for each custom rank.
- MPI_Comm [subGrid_comm](#) [[L_NUM_LEVELS](#) * [L_NUM_REGIONS](#)]
Communicators for sub-grid / region combinations.
- int [my_rank](#)
Rank number.
- int [num_ranks](#)
Total number of ranks in MPI Cartesian topology.
- int [rank_coords](#) [[L_DIMS](#)]
Coordinates in MPI Cartesian topology.
- std::vector< std::vector< double > > [rank_core_edge](#)
Absolute positions of edges of the core region 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_DIRS](#)]
Array of request structures for handles to posted ISends.
- MPI_Status [send_stat](#) [[L_MPI_DIRS](#)]
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.
- std::ofstream * [logout](#)
Logfile handle.

Static Public Attributes

- static const int [neighbour_vectors](#) [3][26]
Cartesian unit vectors pointing to each neighbour in Cartesian topology.

5.17.1 Detailed Description

MPI Manager class.

Class to manage all MPI aspects of the code.

5.17.2 Member Function Documentation

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

Instance destroyer.

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

Instance creator.

5.17.2.3 void `MpiManager::mpi_buffer_pack (int dir, GridObj *const 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 from which information is being sent during the communication.

5.17.2.4 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.

5.17.2.5 void `MpiManager::mpi_buffer_size_recv (GridObj *const 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.
----------	-----------------------

5.17.2.6 void MpiManager::mpi_buffer_size_send (GridObj *const *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.
----------	-----------------------

5.17.2.7 void MpiManager::mpi_buffer_unpack (int *dir*, GridObj *const *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.

5.17.2.8 int MpiManager::mpi_buildCommunicators (GridManager *const *grid_man*)

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.

Parameters

<i>grid_man</i>	pointer to non-null grid manager.
-----------------	-----------------------------------

5.17.2.9 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.

5.17.2.10 `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.
------------------	---

5.17.2.11 `void MpiManager::mpi_gridbuild (GridManager *const grid_man)`

Domain decomposition.

Method to decompose the domain and identify local grid sizes. Parameters defined here are used in [GridObj](#) construction. Grid manager must have been initialised before calling this hence the requirement to have it as a non-null input parameter.

Parameters

<i>grid_man</i>	Pointer to an initialised grid manager.
-----------------	---

5.17.2.12 `void MpiManager::mpi_init ()`

Initialisation routine.

Method is responsible for initialising the MPI topology and associated data. Must be called immediately after `MPI_Init()`. For serial builds this gets called simply to initialise the MPIM with a basic set of grid information used by other methods.

5.17.2.13 `void MpiManager::mpi_updateLoadInfo (GridManager *const grid_man)`

Update the load balancing information stored in the [MpiManager](#).

This method is executed by all processes. Counts the ACTIVE cells on the current rank and pushes the information to the master (rank 0) which writes this information to an output file if required. Must be called after the grids have been built or will return zero.

Parameters

<i>grid_man</i>	pointer to non-null grid manager.
-----------------	-----------------------------------

5.17.2.14 `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.

5.17.3 Member Data Documentation

5.17.3.1 `std::vector<buffer_struct> MpiManager::buffer_rcv_info`

Vectors of buffer_info structures holding receiver layer size info.

5.17.3.2 `std::vector<buffer_struct> MpiManager::buffer_send_info`

Vectors of buffer_info structures holding sender layer size info.

5.17.3.3 `std::vector<int> MpiManager::cRankSizeX`

Number of sites in X direction for each custom rank.

5.17.3.4 `std::vector<int> MpiManager::cRankSizeY`

Number of sites in Y direction for each custom rank.

5.17.3.5 `std::vector<int> MpiManager::cRankSizeZ`

Number of sites in Z direction for each custom rank.

5.17.3.6 `int MpiManager::dimensions[L_DIMS]`

Size of MPI Cartesian topology.

5.17.3.7 `std::vector< std::vector<double> > MpiManager::f_buffer_rcv`

Array of resizable incoming buffers used for data transfer.

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

Array of resizable outgoing buffers used for data transfer.

5.17.3.9 `std::ofstream* MpiManager::logout`

Logfile handle.

5.17.3.10 `int MpiManager::my_rank`

Rank number.

5.17.3.11 `int MpiManager::neighbour_coords[L_DIMS][L_MPI_DIRS]`

Coordinates in MPI topology of neighbour ranks.

5.17.3.12 `int MpiManager::neighbour_rank[L_MPI_DIRS]`

Neighbour rank number for each direction in Cartesian topology.

5.17.3.13 `const int MpiManager::neighbour_vectors [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. MSVC 2013 does not support initialiser lists tagged onto the constructor although it is valid C++ so I have had to make it static even though it goes against the idea of the singleton design.

5.17.3.14 `int MpiManager::num_ranks`

Total number of ranks in MPI Cartesian topology.

5.17.3.15 `int MpiManager::rank_coords[L_DIMS]`

Coordinates in MPI Cartesian topology.

5.17.3.16 `std::vector< std::vector<double> > MpiManager::rank_core_edge`

Absolute positions of edges of the core region represented on this rank.

Excludes outer overlapping layer (recv layer). Rows are x,y,z start and end pairs and columns are rank number. Access the rows using the eCartMinMax enumeration.

5.17.3.17 `layer_edges` `MpiManager::recv_layer_pos`

Structure containing receiver layer edge positions.

5.17.3.18 `MPI_Status` `MpiManager::recv_stat`

Status structure for Receive return information.

5.17.3.19 `MPI_Request` `MpiManager::send_requests[L_MPI_DIRS]`

Array of request structures for handles to posted ISends.

5.17.3.20 `MPI_Status` `MpiManager::send_stat[L_MPI_DIRS]`

Array of statuses for each Isend.

5.17.3.21 `layer_edges` `MpiManager::sender_layer_pos`

Structure containing sender layer edge positions.

5.17.3.22 `MPI_Comm` `MpiManager::subGrid_comm[L_NUM_LEVELS * L_NUM_REGIONS]`

Communicators for sub-grid / region combinations.

5.17.3.23 `MPI_Comm` `MpiManager::world_comm`

Global MPI communicator.

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

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

5.18 ObjectManager Class Reference

Object Manager class.

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

Public Member Functions

- void [ibm_apply](#) ()
Perform IBM procedure.
- void [ibm_buildBody](#) (int body_type)
Builds a prefab immersed boundary body.
- void [ibm_buildBody](#) (PCpts *_PCpts, GridObj *owner)
Wrapper for building a body from a point cloud.
- void [ibm_initialise](#) ()
Initialise the array of iBodies.
- double [ibm_deltaKernel](#) (double rad, double dilation)
Method to evaluate delta kernel at supplied location.
- void [ibm_interpol](#) (int ib)
Interpolate velocity field onto markers.
- void [ibm_spread](#) (int ib)
Spread restorative force back onto marker support.
- void [ibm_findSupport](#) (int ib, int m)
Finds support points for iBody.
- void [ibm_initialiseSupport](#) (int ib, int m, int s, double estimated_position[])
Initialise data associated with support points found.
- void [ibm_computeForce](#) (int ib)
Compute restorative force at each marker in a body.
- double [ibm_findEpsilon](#) (int ib)
Compute epsilon for a given iBody.
- void [ibm_moveBodies](#) ()
Moves iBodies after applying IBM.
- double [ibm_bicgstab](#) (std::vector< std::vector< double > > &Amatrix, std::vector< double > &bVector, std::vector< double > &epsilon, double tolerance, int maxiterations)
Biconjugate gradient method.
- void [ibm_jacowire](#) (int ib)
Structural calculation of flexible cilia.
- void [ibm_positionUpdate](#) (int ib)
Update the position of a movable iBody.
- void [ibm_positionUpdateGroup](#) (int group)
Update the position of a group of movable iBodies.
- void [ibm_banbks](#) (double **a, long n, int m1, int m2, double **al, unsigned long indx[], double b[])
Solution of a banded diagonal linear system.
- void [ibm_bandec](#) (double **a, long n, int m1, int m2, double **al, unsigned long indx[], double *d)
LU decomposition of band diagonal matrix.
- void [bfl_buildBody](#) (int body_type)
Prefab body building routine.
- void [bfl_buildBody](#) (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_vtkIBBWriter](#) (double tval)
Write IB body data to VTK file.
- void [io_writeBodyPosition](#) (int timestep)
Write out position of immersed boundary bodies.
- void [io_writeLiftDrag](#) (int timestep)
Write out forces on the markers of immersed boundary bodies.

- void [io_restart](#) ([eIOFlag](#) 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](#)

5.18.1 Detailed Description

Object Manager class.

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

5.18.2 Member Function Documentation

5.18.2.1 void ObjectManager::bfl_buildBody (int *body_type*)

Prefab body building routine.

Not implemented in this version.

Parameters

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

5.18.2.2 void ObjectManager::bfl_buildBody ([PCpts](#) * _PCpts)

Wrapper for building BFL body from point cloud.

Parameters

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

5.18.2.3 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.

5.18.2.4 void ObjectManager::destroyInstance () [static]

Destroy instance method.

Instance destructor.

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

Get instance method.

Instance creator.

5.18.2.6 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.
----------	----------------------------

5.18.2.7 void ObjectManager::ibm_apply ()

Perform IBM procedure.

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

Solution of a banded diagonal linear system.

Given the arrays *A*, *AL*, and *INDX* as returned from [ibm_banded\(\)](#), and given a right-hand side vector *B*[1..*n*], solves the band diagonal linear equations $AX = B$. The solution vector *X* overwrites *B*. The other input arrays are not modified, and can be left in place for successive calls with different right-hand sides. (C) Copr. 1986-92 Numerical Recipes Software ?421.1-9.

Parameters

<i>a</i>	array of subdiagonal and superdiagonals rows
<i>n</i>	size of the square matrix <i>A</i>
<i>m1</i>	number of subdiagonal rows
<i>m2</i>	number of superdiagonal rows
<i>al</i>	lower triangular matrix
<i>indx</i>	row permutation vector
<i>b</i>	right hand side vector

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

LU decomposition of band diagonal matrix.

Given an *n* by *n* band diagonal matrix *A* with *m1* subdiagonal rows and *m2* superdiagonal rows, compactly stored in the array *A*[1..*n*][1..*m1*+*m2*+1], this routine constructs an LU decomposition of a rowwise permutation of *A*. The upper triangular matrix replaces *A*, while the lower triangular matrix is returned in *AL*[1..*n*][1..*m1*]. *INDX*[1..*n*] is an output vector which records the row permutation effected by the partial pivoting; *D* is output as +/-1 depending on whether the number of row interchanges was even or odd, respectively. This routine is used in combination with [ibm_banbks\(\)](#) to solve band-diagonal sets of equations. Once the matrix *A* has been decomposed, any number of right-hand sides can be solved in turn by repeated calls to [ibm_banbks\(\)](#). (C) Copr. 1986-92 Numerical Recipes Software ?421.1-9.

Parameters

<i>a</i>	array of subdiagonal and superdiagonals rows
<i>n</i>	size of the square matrix <i>A</i>
<i>m1</i>	number of subdiagonal rows
<i>m2</i>	number of superdiagonal rows
<i>al</i>	lower triangular matrix
<i>indx</i>	row permutation vector
<i>d</i>	odd or even number of row interchanges

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

Biconjugate gradient method.

Biconjugate gradient stabilised method of solving a linear system $Ax = b$. Solution is performed iteratively.

Parameters

<i>Amatrix</i>	the <i>A</i> matrix in the linear system.
----------------	---

Parameters

<i>bVector</i>	the b vector in the linear system.
<i>epsilon</i>	epsilon paramters for each marker.
<i>tolerance</i>	tolerance of solution.
<i>maxiterations</i>	maximum number of iterations.

Returns

the minimum residual achieved by the solver.

5.18.2.11 void ObjectManager::ibm_buildBody (int *body_type*)

Builds a prefab immersed boundary body.

Parameters

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

5.18.2.12 void ObjectManager::ibm_buildBody (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.

5.18.2.13 void ObjectManager::ibm_computeForce (int *ib*)

Compute restorative force at each marker in a body.

Parameters

<i>ib</i>	iBody being operated on.
-----------	--------------------------

5.18.2.14 double ObjectManager::ibm_deltaKernel (double *radius*, double *dilation*)

Method to evaluate delta kernel at supplied location.

Radius and dilation must be in the same units.

Parameters

<i>radius</i>	location at which kernel should be evaluated.
<i>dilation</i>	width of kernel function.

Returns

value of kernel function.

5.18.2.15 double ObjectManager::ibm_findEpsilon (int *ib*)

Compute epsilon for a given iBody.

Parameters

<i>ib</i>	iBody being operated on.
-----------	--------------------------

5.18.2.16 void ObjectManager::ibm_findSupport (int *ib*, int *m*)

Finds support points for iBody.

Support for given marker in given body is sought on the owning grid.

Parameters

<i>ib</i>	body under consideration.
<i>m</i>	marker whose support is to be found.

5.18.2.17 void ObjectManager::ibm_initialise ()

Initialise the array of iBodies.

Computes support and epsilon values.

5.18.2.18 void ObjectManager::ibm_initialiseSupport (int *ib*, int *m*, int *s*, double *estimated_position*[])

Initialise data associated with support points found.

Finds and stores the delta values of the support points.

Parameters

<i>ib</i>	iBody being operated on.
<i>m</i>	marker of interest.
<i>s</i>	support point of interest.
<i>estimated_position</i>	vector containing the estimated position of the support point.

5.18.2.19 void ObjectManager::ibm_interpol (int *ib*)

Interpolate velocity field onto markers.

Parameters

<i>ib</i>	iBody being operated on.
-----------	--------------------------

5.18.2.20 void ObjectManager::ibm_jacowire (int *ib*)

Structural calculation of flexible cilia.

Models the structural behaviour of a thin wire using Euler-Bernoulli beam elements. Only implemented for one simply supported end and one free end at present.

Parameters

<i>ib</i>	index of body to which calculation is to be applied.
-----------	--

5.18.2.21 void ObjectManager::ibm_moveBodies ()

Moves iBodies after applying IBM.

Wrapper for relocating markers of an iBody by calling appropriate positional update routine.

5.18.2.22 void ObjectManager::ibm_positionUpdate (int *ib*)

Update the position of a movable iBody.

Wrapper for applying external forcing or structural calculations to iBodies marked as movable. Updates support on completion.

Parameters

<i>ib</i>	index of body to which calculation is to be applied.
-----------	--

5.18.2.23 void ObjectManager::ibm_positionUpdateGroup (int *group*)

Update the position of a group of movable iBodies.

Updates the position of a group of non-flexible movable bodies by using the first flexible body in the group as the driver. Must be called after all previous positional update routines have been called.

Parameters

<i>group</i>	group ID to be updated.
--------------	-------------------------

5.18.2.24 void ObjectManager::ibm_spread (int *ib*)

Spread restorative force back onto marker support.

Parameters

<i>ib</i>	iBody being operated on.
-----------	--------------------------

5.18.2.25 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.

5.18.2.26 void ObjectManager::io_restart (eIOFlag *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

5.18.2.27 void ObjectManager::io_vtkIBBWriter (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.
-------------	---

5.18.2.28 void ObjectManager::io_writeBodyPosition (int *timestep*)

Write out position of immersed boundary bodies.

Parameters

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

5.18.2.29 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.
-------------	--

5.18.2.30 void ObjectManager::io_writeLiftDrag (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.
-----------------	---

5.18.3 Friends And Related Function Documentation

5.18.3.1 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_ibmbody.cpp](#)
- [ObjectManager_ops_ibm.cpp](#)
- [ObjectManager_ops_ibmflex.cpp](#)
- [ObjectManager_ops_io.cpp](#)

5.19 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.

5.19.1 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.

5.19.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

5.19.3 Member Data Documentation

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

Vector of X positions.

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

Vector of Y positions.

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

Vector of Z positions.

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

- [PCpts.h](#)

Chapter 6

File Documentation

6.1 BFLBody.cpp File Reference

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

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"
#include "MarkerData.h"
```

Classes

- 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 "1.4.0-alpha"`
LUMA version.
- `#define L_INIT_VERBOSE`
Write out initialisation information such as refinement mappings.
- `#define L_MPI_VERBOSE`
Write out the buffers used by MPI plus more setup data.
- `#define L_MPI_WRITE_LOAD_BALANCE`
Write out the load balancing information based on active cell count.
- `#define L_CLOUD_DEBUG`
Write out to a file the cloud that has been read in.
- `#define L_LOG_TIMINGS`
Write out the initialisation, time step and mpi timings to an output file.
- `#define L_HDF_DEBUG`
Write some HDF5 debugging information.
- `#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 1`
Specific output frequency of body forces.
- `#define L_OUTPUT_PRECISION 5`
Precision of output (for text writers)
- `#define L_HDF5_OUTPUT`
HDF5 dump on output.
- `#define L_LD_OUT`
Write out lift and drag (all bodies)
- `#define L_PROBE_OUT_FREQ 250`
Write out frequency of probe output.
- `#define L_GRAVITY_FORCE 0.0001`
Expression for the gravity force.
- `#define L_GRAVITY_DIRECTION eXDirection`
Gravity direction (specify using enumeration)
- `#define L_RESTART_OUT_FREQ 1000`
Frequency of write out of restart file.
- `#define L_CSMAG 0.07`
- `#define L_TOTAL_TIMESTEPS 1000`
Number of time steps to run simulation for.
- `#define L_MPI_XCORES 2`
Number of MPI ranks to divide domain into in X direction.
- `#define L_MPI_YCORES 2`
- `#define L_MPI_ZCORES 2`
Number of MPI ranks to divide domain into in Z direction.
- `#define L_DIMS 2`
Number of dimensions to the problem.
- `#define L_RESOLUTION 40`
Number of coarse lattice sites per unit length.
- `#define L_TIMESTEP 0.1`
The timestep in non-dimensional units.
- `#define L_BX 10`
End of domain in X (non-dimensional units)
- `#define L_BY 10`
End of domain in Y (non-dimensional units)
- `#define L_BZ 10`
End of domain in Z (non-dimensional units)
- `#define L_PHYSICAL_U 0.2`
Reference velocity of the real fluid to model [m/s].
- `#define L_UREF 0.04`
Reference velocity for scaling.
- `#define L_UMAX L_UREF*1.5`
Max velocity of inlet profile.
- `#define L_UX0 0.04`
Initial/inlet x-velocity.
- `#define L_UY0 0.0`
Initial/inlet y-velocity.
- `#define L_UZ0 0.0`

- Initial/inlet z-velocity.*

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

 - `#define L_RE 150`
- Desired Reynolds number.*

 - `#define L_IB_ON_LEV 0`
- Grid level for immersed boundary object (0 if no refined regions, -1 if no IBM)*

 - `#define L_IB_ON_REG 0`
- Grid region for immersed boundary object (0 if no refined regions, -1 if no IBM)*

 - `#define L_VTK_BODY_WRITE`
- Write out the bodies to a VTK file.*

 - `#define L_IBB_FROM_FILE`
- Build immersed bodies from a point cloud file.*

 - `#define L_IBB_ON_GRID_LEV L_IB_ON_LEV`
- Provide grid level on which object should be added.*

 - `#define L_IBB_ON_GRID_REG L_IB_ON_REG`
- Provide grid region on which object should be added.*

 - `#define L_START_IBB_X 0.9`
- Start X of object bounding box.*

 - `#define L_START_IBB_Y 0.4`
- 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 0.2`
- 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_REF_LENGTH 0.2`
- Reference length to be used in the definition of Reynolds number.*

 - `#define L_NUM_MARKERS 31`
- Number of Lagrange points to use when building a prefab body (approximately)*

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

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

 - `#define L_IBB_X 0.2`
- X Position of body centre.*

 - `#define L_IBB_Y 0.2`
- Y Position of body centre.*

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

 - `#define L_IBB_W 0.5`
- Width (x) of IB body.*

 - `#define L_IBB_L 0.5`
- Length (y) of IB body.*

 - `#define L_IBB_D 0.5`
- Depth (z) of IB body.*

 - `#define L_IBB_R 0.05`
- Radius of IB body.*

 - `#define L_IBB_FILAMENT_LENGTH 0.5`
- Length of filament.*

- `#define L_IBB_FILAMENT_START_X 0.2`
Start X position of the filament.
- `#define L_IBB_FILAMENT_START_Y 0.5`
Start Y position of the filament.
- `#define L_IBB_FILAMENT_START_Z 0.5`
Start Z position of the filament.
- `#define L_IBB_ANGLE_VERT 90`
Inclination of filament in XY plane.
- `#define L_IBB_ANGLE_HORZ 0`
Inclination of filament in XZ plane.
- `#define L_FILAMENT_START_BC 2`
Type of boundary condition at filament start: 0 == free; 1 = simply supported; 2 == clamped.
- `#define L_FILAMENT_END_BC 0`
Type of boundary condition at filament end: 0 == free; 1 = simply supported; 2 == clamped.
- `#define L_IBB_DELTA_RHO 1.0`
Difference in density (lattice units) between solid and fluid.
- `#define L_IBB_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 Do Nothing)
- `#define L_PERIODIC_BOUNDARIES`
Turn on periodic boundary conditions (doesn't do anything anymore – periodic by default)
- `#define L_WALL_THICKNESS_BOTTOM (static_cast<double>(L_BX)/static_cast<double>(L_N))`
Thickness of wall.
- `#define L_WALL_THICKNESS_TOP (static_cast<double>(L_BX)/static_cast<double>(L_N))`
Thickness of top wall.
- `#define L_WALL_THICKNESS_FRONT (static_cast<double>(L_BX)/static_cast<double>(L_N))`
Thickness of front (3D) wall.
- `#define L_WALL_THICKNESS_BACK (static_cast<double>(L_BX)/static_cast<double>(L_N))`
Thickness of back (3D) wall.
- `#define L_BLOCK_ON_GRID_LEV 2`
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_MIN_X 0.9`
Start of object/wall in x-direction.
- `#define L_BLOCK_MAX_X 1.1`
End of object/wall in x-direction.
- `#define L_BLOCK_MIN_Y 0.4`
Start of object/wall in y-direction.
- `#define L_BLOCK_MAX_Y 0.6`
End of object/wall in y-direction.
- `#define L_BLOCK_MIN_Z 0.3`
Start of object/wall in z-direction.
- `#define L_BLOCK_MAX_Z 0.7`
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` 2
 - 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` (9.06 / 2.0)
 - Start of object bounding box in X direction.*
 - #define `L_START_OBJECT_Y` (9.66 / 2.0)
 - Start of object bounding box in Y direction.*
 - #define `L_CENTRE_OBJECT_Z` 5.0
 - Centre of object bounding box in Z direction.*
 - #define `L_OBJECT_LENGTH` 0.94
 - 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_REF_LENGTH` 1.0
 - Reference length to be used in the definition of Reynolds number.*
 - #define `L_BFL_ON_GRID_LEV` 2
 - 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` 0.9
 - Start of object bounding box in X direction.*
 - #define `L_START_BFL_Y` 0.4
 - Start of object bounding box in Y direction.*
 - #define `L_CENTRE_BFL_Z` 0.5
 - Centre of object bounding box in Z direction.*
 - #define `L_BFL_LENGTH` 0.2
 - 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_REF_LENGTH` 0.2
 - Reference length to be used in the definition of Reynolds number.*
 - #define `L_NUM_LEVELS` 2
 - Levels of refinement (0 = coarse grid only)*
 - #define `L_NUM_REGIONS` 1
 - Number of refined regions (can be arbitrary if L_NUM_LEVELS = 0)*
 - #define `L_N` `static_cast<int>(L_BX * L_RESOLUTION)`
 - #define `L_M` `static_cast<int>(L_BY * L_RESOLUTION)`
 - #define `L_K` `static_cast<int>(L_BZ * L_RESOLUTION)`
 - #define `L_NUM_VELS` 9
 - #define `L_MPI_DIRS` 8
 - #define `L_BZ` 2
 - End of domain in Z (non-dimensional units)*
 - #define `L_K` 1
 - #define `L_MPI_ZCORES` 1
 - Number of MPI ranks to divide domain into in Z direction.*
 - #define `L_BLOCK_MIN_Z` 0.0
 - Start of object/wall in z-direction.*
 - #define `L_BLOCK_MAX_Z` 0.0
 - End of object/wall in z-direction.*

- `#define L_IBB_D 0.0`
Depth (z) of IB body.
- `#define L_CENTRE_OBJECT_Z 0.0`
Centre of object bounding box in Z direction.
- `#define L_CENTRE_BFL_Z 0.0`
Centre of object bounding box in Z direction.
- `#define L_CENTRE_IBB_Z 0.0`
Centre of object bounding box in Z direction.
- `#define L_UZ0 0.0`
Initial/inlet z-velocity.

Variables

- static const int `cNumProbes [3] = {3, 3, 3}`
Number of probes in each direction (x, y, z)
- static const double `cProbeLimsX [2] = {0.1, 0.2}`
Limits of X plane for array of probes.
- static const double `cProbeLimsY [2] = {0.1, 0.2}`
Limits of Y plane for array of probes.
- static const double `cProbeLimsZ [2] = {0.1, 0.2}`
Limits of Z plane for array of probes.
- static double `cRefStartX [L_NUM_LEVELS][L_NUM_REGIONS]`
- static double `cRefEndX [L_NUM_LEVELS][L_NUM_REGIONS]`
- static double `cRefStartY [L_NUM_LEVELS][L_NUM_REGIONS]`
- static double `cRefEndY [L_NUM_LEVELS][L_NUM_REGIONS]`
- static double `cRefStartZ [L_NUM_LEVELS][L_NUM_REGIONS]`
- static double `cRefEndZ [L_NUM_LEVELS][L_NUM_REGIONS]`

6.6.1 Macro Definition Documentation

6.6.1.1 `#define L_BFL_LENGTH 0.2`

The BFL object input is scaled based on this dimension.

6.6.1.2 `#define L_BFL_ON_GRID_LEV 2`

Provide grid level on which BFL body should be added.

6.6.1.3 `#define L_BFL_ON_GRID_REG 0`

Provide grid region on which BFL body should be added.

6.6.1.4 `#define L_BFL_REF_LENGTH 0.2`

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

6.6.1.5 `#define L_BFL_SCALE_DIRECTION eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.6 `#define L_BLOCK_MAX_X 1.1`

End of object/wall in x-direction.

6.6.1.7 `#define L_BLOCK_MAX_Y 0.6`

End of object/wall in y-direction.

6.6.1.8 `#define L_BLOCK_MAX_Z 0.7`

End of object/wall in z-direction.

6.6.1.9 `#define L_BLOCK_MAX_Z 0.0`

End of object/wall in z-direction.

6.6.1.10 `#define L_BLOCK_MIN_X 0.9`

Start of object/wall in x-direction.

6.6.1.11 `#define L_BLOCK_MIN_Y 0.4`

Start of object/wall in y-direction.

6.6.1.12 `#define L_BLOCK_MIN_Z 0.3`

Start of object/wall in z-direction.

6.6.1.13 `#define L_BLOCK_MIN_Z 0.0`

Start of object/wall in z-direction.

6.6.1.14 `#define L_BLOCK_ON_GRID_LEV 2`

Provide grid level on which block should be added.

6.6.1.15 #define L_BLOCK_ON_GRID_REG 0

Provide grid region on which block should be added.

6.6.1.16 #define L_BUILD_FOR_MPI

Enable MPI features in build.

6.6.1.17 #define L_BX 10

End of domain in X (non-dimensional units)

6.6.1.18 #define L_BY 10

End of domain in Y (non-dimensional units)

6.6.1.19 #define L_BZ 10

End of domain in Z (non-dimensional units)

6.6.1.20 #define L_BZ 2

End of domain in Z (non-dimensional units)

6.6.1.21 #define L_CENTRE_BFL_Z 0.5

Centre of object bounding box in Z direction.

6.6.1.22 #define L_CENTRE_BFL_Z 0.0

Centre of object bounding box in Z direction.

6.6.1.23 #define L_CENTRE_IBB_Z 0.5

Centre of object bounding box in Z direction.

6.6.1.24 #define L_CENTRE_IBB_Z 0.0

Centre of object bounding box in Z direction.

6.6.1.25 `#define L_CENTRE_OBJECT_Z 5.0`

Centre of object bounding box in Z direction.

6.6.1.26 `#define L_CENTRE_OBJECT_Z 0.0`

Centre of object bounding box in Z direction.

6.6.1.27 `#define L_CLOUD_DEBUG`

Write out to a file the cloud that has been read in.

6.6.1.28 `#define L_CSMAG 0.07`

6.6.1.29 `#define L_DIMS 2`

Number of dimensions to the problem.

6.6.1.30 `#define L_FILAMENT_END_BC 0`

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

6.6.1.31 `#define L_FILAMENT_START_BC 2`

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

6.6.1.32 `#define L_FREESTREAM_TUNNEL`

Adds a inlet to all faces.

6.6.1.33 `#define L_GRAVITY_DIRECTION eXDirection`

Gravity direction (specify using enumeration)

6.6.1.34 `#define L_GRAVITY_FORCE 0.0001`

Expression for the gravity force.

6.6.1.35 `#define L_HDF5_OUTPUT`

HDF5 dump on output.

6.6.1.36 #define L_HDF_DEBUG

Write some HDF5 debugging information.

6.6.1.37 #define L_IB_ON_LEV 0

Grid level for immersed boundary object (0 if no refined regions, -1 if no IBM)

6.6.1.38 #define L_IB_ON_REG 0

Grid region for immersed boundary object (0 if no refined regions, -1 if no IBM)

6.6.1.39 #define L_IBB_ANGLE_HORZ 0

Inclination of filament in XZ plane.

6.6.1.40 #define L_IBB_ANGLE_VERT 90

Inclination of filament in XY plane.

6.6.1.41 #define L_IBB_D 0.5

Depth (z) of IB body.

6.6.1.42 #define L_IBB_D 0.0

Depth (z) of IB body.

6.6.1.43 #define L_IBB_DELTA_RHO 1.0

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

6.6.1.44 #define L_IBB_EI 2.0

Flexural rigidity (lattice units) of filament.

6.6.1.45 #define L_IBB_FILAMENT_LENGTH 0.5

Length of filament.

6.6.1.46 `#define L_IBB_FILAMENT_START_X 0.2`

Start X position of the filament.

6.6.1.47 `#define L_IBB_FILAMENT_START_Y 0.5`

Start Y position of the filament.

6.6.1.48 `#define L_IBB_FILAMENT_START_Z 0.5`

Start Z position of the filament.

6.6.1.49 `#define L_IBB_FLEXIBLE false`

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

6.6.1.50 `#define L_IBB_FROM_FILE`

Build immersed bodies from a point cloud file.

6.6.1.51 `#define L_IBB_L 0.5`

Length (y) of IB body.

6.6.1.52 `#define L_IBB_LENGTH 0.2`

The object input is scaled based on this dimension.

6.6.1.53 `#define L_IBB_MOVABLE false`

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

6.6.1.54 `#define L_IBB_ON_GRID_LEV L_IB_ON_LEV`

Provide grid level on which object should be added.

6.6.1.55 `#define L_IBB_ON_GRID_REG L_IB_ON_REG`

Provide grid region on which object should be added.

6.6.1.56 `#define L_IBB_R 0.05`

Radius of IB body.

6.6.1.57 `#define L_IBB_REF_LENGTH 0.2`

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

6.6.1.58 `#define L_IBB_SCALE_DIRECTION eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.59 `#define L_IBB_W 0.5`

Width (x) of IB body.

6.6.1.60 `#define L_IBB_X 0.2`

X Position of body centre.

6.6.1.61 `#define L_IBB_Y 0.2`

Y Position of body centre.

6.6.1.62 `#define L_IBB_Z 0.0`

Z Position of body centre.

6.6.1.63 `#define L_INIT_VERBOSE`

Write out initialisation information such as refinement mappings.

6.6.1.64 `#define L_INLET_ON`

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

6.6.1.65 `#define L_K static_cast<int>(L_BZ * L_RESOLUTION)`

6.6.1.66 `#define L_K 1`

6.6.1.67 `#define L_LD_OUT`

Write out lift and drag (all bodies)

6.6.1.68 `#define L_LOG_TIMINGS`

Write out the initialisation, time step and mpi timings to an output file.

6.6.1.69 `#define L_M static_cast<int>(L_BY * L_RESOLUTION)`

6.6.1.70 `#define L_MPI_DIRS 8`

6.6.1.71 `#define L_MPI_VERBOSE`

Write out the buffers used by MPI plus more setup data.

6.6.1.72 `#define L_MPI_WRITE_LOAD_BALANCE`

Write out the load balancing information based on active cell count.

6.6.1.73 `#define L_MPI_XCORES 2`

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

6.6.1.74 `#define L_MPI_YCORES 2`

Number of MPI ranks to divide domain into in Y direction

6.6.1.75 `#define L_MPI_ZCORES 2`

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

6.6.1.76 `#define L_MPI_ZCORES 1`

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

6.6.1.77 `#define L_N static_cast<int>(L_BX * L_RESOLUTION)`

6.6.1.78 `#define L_NUM_LEVELS 2`

Levels of refinement (0 = coarse grid only)

6.6.1.79 `#define L_NUM_MARKERS 31`

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

6.6.1.80 #define L_NUM_REGIONS 1

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

6.6.1.81 #define L_NUM_VELS 9**6.6.1.82 #define L_OBJECT_LENGTH 0.94**

The object input is scaled based on this dimension.

6.6.1.83 #define L_OBJECT_ON_GRID_LEV 2

Provide grid level on which object should be added.

6.6.1.84 #define L_OBJECT_ON_GRID_REG 0

Provide grid region on which object should be added.

6.6.1.85 #define L_OBJECT_REF_LENGTH 1.0

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

6.6.1.86 #define L_OBJECT_SCALE_DIRECTION eXDirection

Scale in this direction (specify as enumeration)

6.6.1.87 #define L_OUT_EVERY 100

How many timesteps before whole grid output.

6.6.1.88 #define L_OUT_EVERY_FORCES 1

Specific output frequency of body forces.

6.6.1.89 #define L_OUTLET_ON

Turn on outlet boundary (assumed right-hand wall – default Do Nothing)

6.6.1.90 #define L_OUTPUT_PRECISION 5

Precision of output (for text writers)

6.6.1.91 #define L_PERIODIC_BOUNDARIES

Turn on periodic boundary conditions (doesn't do anything anymore – periodic by default)

6.6.1.92 #define L_PHYSICAL_U 0.2

Reference velocity of the real fluid to model [m/s].

6.6.1.93 #define L_PI 3.14159265358979323846

PI definition.

6.6.1.94 #define L_PROBE_OUT_FREQ 250

Write out frequency of probe output.

6.6.1.95 #define L_RE 150

Desired Reynolds number.

6.6.1.96 #define L_RESOLUTION 40

Number of coarse lattice sites per unit length.

6.6.1.97 #define L_RESTART_OUT_FREQ 1000

Frequency of write out of restart file.

6.6.1.98 #define L_RHOIN 1

Initial density.

6.6.1.99 #define L_SOLID_FROM_FILE

Build solid body from point cloud file.

6.6.1.100 #define L_START_BFL_X 0.9

Start of object bounding box in X direction.

6.6.1.101 `#define L_START_BFL_Y 0.4`

Start of object bounding box in Y direction.

6.6.1.102 `#define L_START_IBB_X 0.9`

Start X of object bounding box.

6.6.1.103 `#define L_START_IBB_Y 0.4`

Start Y of object bounding box.

6.6.1.104 `#define L_START_OBJECT_X (9.06 / 2.0)`

Start of object bounding box in X direction.

6.6.1.105 `#define L_START_OBJECT_Y (9.66 / 2.0)`

Start of object bounding box in Y direction.

6.6.1.106 `#define L_TIMESTEP 0.1`

The timestep in non-dimensional units.

6.6.1.107 `#define L_TOTAL_TIMESTEPS 1000`

Number of time steps to run simulation for.

6.6.1.108 `#define L_UMAX L_UREF*1.5`

Max velocity of inlet profile.

6.6.1.109 `#define L_UREF 0.04`

Reference velocity for scaling.

6.6.1.110 `#define L_UX0 0.04`

Initial/inlet x-velocity.

6.6.1.111 `#define L_UY0 0.0`

Initial/inlet y-velocity.

6.6.1.112 `#define L_UZ0 0.0`

Initial/inlet z-velocity.

6.6.1.113 `#define L_UZ0 0.0`

Initial/inlet z-velocity.

6.6.1.114 `#define L_VTK_BODY_WRITE`

Write out the bodies to a VTK file.

6.6.1.115 `#define L_WALL_THICKNESS_BACK (static_cast<double>(L_BX)/static_cast<double>(L_N))`

Thickness of back (3D) wall.

6.6.1.116 `#define L_WALL_THICKNESS_BOTTOM (static_cast<double>(L_BX)/static_cast<double>(L_N))`

Thickness of wall.

6.6.1.117 `#define L_WALL_THICKNESS_FRONT (static_cast<double>(L_BX)/static_cast<double>(L_N))`

Thickness of front (3D) wall.

6.6.1.118 `#define L_WALL_THICKNESS_TOP (static_cast<double>(L_BX)/static_cast<double>(L_N))`

Thickness of top wall.

6.6.1.119 `#define LUMA_VERSION "1.4.0-alpha"`

LUMA version.

6.6.2 Variable Documentation

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

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

6.6.2.2 `const double cProbeLimsX[2] = {0.1, 0.2}` `[static]`

Limits of X plane for array of probes.

6.6.2.3 `const double cProbeLimsY[2] = {0.1, 0.2}` `[static]`

Limits of Y plane for array of probes.

6.6.2.4 `const double cProbeLimsZ[2] = {0.1, 0.2}` `[static]`

Limits of Z plane for array of probes.

6.6.2.5 `double cRefEndX[L_NUM_LEVELS][L_NUM_REGIONS]` `[static]`

Initial value:

```
= {  
    { 6.5 },  
    { 6.0 }  
}
```

6.6.2.6 `double cRefEndY[L_NUM_LEVELS][L_NUM_REGIONS]` `[static]`

Initial value:

```
= {  
    { 6.5 },  
    { 6.0 }  
}
```

6.6.2.7 `double cRefEndZ[L_NUM_LEVELS][L_NUM_REGIONS]` `[static]`

Initial value:

```
= {  
    { 6.5 },  
    { 6.0 }  
}
```

6.6.2.8 `double cRefStartX[L_NUM_LEVELS][L_NUM_REGIONS]` `[static]`

Initial value:

```
= {  
    { 3.5 },  
    { 4.0 }  
}
```

6.6.2.9 double cRefStartY[L_NUM_LEVELS][L_NUM_REGIONS] [static]

Initial value:

```
= {
    { 3.5 },
    { 4.0 }
}
```

6.6.2.10 double cRefStartZ[L_NUM_LEVELS][L_NUM_REGIONS] [static]

Initial value:

```
= {
    { 3.5 },
    { 4.0 }
}
```

6.7 Enumerations.h File Reference

Enumerations

- enum [eCartesianDirection](#) { [eXDirection](#), [eYDirection](#), [eZDirection](#) }
Enumeration for directional options.
- enum [eCartMinMax](#) {
[eXMin](#), [eXMax](#), [eYMin](#), [eYMax](#),
[eZMin](#), [eZMax](#) }
Enumeration for the combination of eCartesianDirection and eMinMax as these are often used together to index arrays.
- enum [eLocationOnRank](#) { [eNone](#), [eCore](#), [eHalo](#) }
Enumeration indicating the location of a site when queried using isOnThisRank()
- enum [eMinMax](#) { [eMinimum](#), [eMaximum](#) }
Enumeration for minimum and maximum.
- enum [eEdgeMinMax](#) { [eLeftMin](#), [eLeftMax](#), [eRightMin](#), [eRightMax](#) }
Enumeration for the combination of Left and Right min and max edges.
- enum [eIBInfoType](#) {
[eIBDeltaSum](#), [eIBEpsilon](#), [eIBVelocityInterpolation](#), [eIBVelocitySpreading](#),
[eIBMarkerPositions](#) }
Type of container required.
- 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.
- enum [eIOFlag](#) { [eWrite](#), [eRead](#) }
Flag for indicating write or read action for IO methods.
- enum [eObjectType](#) { [eBBBCloud](#), [eBFLCloud](#), [eIBBCloud](#) }
Specifies the type of body being processed.
- enum [eHdf5SlabType](#) {
[eScalar](#), [eVector](#), [eProductVector](#), [ePosX](#),
[ePosY](#), [ePosZ](#) }
Defines the type of storage arrangement of the variable in memory.

6.7.1 Enumeration Type Documentation

6.7.1.1 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.

6.7.1.2 enum eCartesianDirection

Enumeration for directional options.

Enumerator

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

6.7.1.3 enum eCartMinMax

Enumeration for the combination of eCartesianDirection and eMinMax as these are often used together to index arrays.

Enumerator

- eXMin**
- eXMax**
- eYMin**
- eYMax**
- eZMin**
- eZMax**

6.7.1.4 enum eEdgeMinMax

Enumeration for the combination of Left and Right min and max edges.

Enumerator

- eLeftMin**
- eLeftMax**
- eRightMin**
- eRightMax**

6.7.1.5 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

6.7.1.6 enum eBInfoType

Type of container required.

Enumerator

- eBDeltaSum**
- eBEpsilon**
- eBVelocityInterpolation**
- eBVelocitySpreading**
- eBMarkerPositions**

6.7.1.7 enum eOFlag

Flag for indicating write or read action for IO methods.

Enumerator

- eWrite** Write to file.
- eRead** Read from file.

6.7.1.8 enum eLocationOnRank

Enumeration indicating the location of a site when queried using isOnThisRank()

Enumerator

- eNone** No information provided (default).
- eCore** Site on core (including send layer).
- eHalo** Site in halo (recv layer).

6.7.1.9 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.

6.7.1.10 enum **eObjectType**

Specifies the type of body being processed.

Enumerator

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

6.7.1.11 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.

6.8 GridManager.cpp File Reference

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

6.9 GridManager.h File Reference

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

Classes

- class [GridManager](#)
Grid Manager class.

6.10 GridObj.cpp File Reference

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

6.11 GridObj.h File Reference

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

Classes

- class [GridObj](#)
Grid class.

6.12 GridObj_init_grids.cpp File Reference

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

6.13 GridObj_ops_boundary.cpp File Reference

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

6.14 GridObj_ops_io.cpp File Reference

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

6.15 GridObj_ops_lbm.cpp File Reference

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

6.16 GridObj_ops_lbm_optimised.cpp File Reference

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

6.17 GridUnits.h File Reference

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

Classes

- class [GridUnits](#)
[GridUnits.](#)

6.18 GridUtils.cpp File Reference

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

6.19 GridUtils.h File Reference

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

Classes

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

6.20 hdf5luma.h File Reference

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

Macros

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

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, [GridObj](#) *g, T *data, hid_t hdf_datatype, bool *TL_present, int TL_thickness, [HDFstruct](#) hdf_data)
Helper method to write out using HDF5.

6.20.1 Macro Definition Documentation

6.20.1.1 #define [H5_BUILT_AS_DYNAMIC_LIB](#)

6.20.1.2 #define [HDF5_EXT_SZIP](#)

6.20.1.3 #define [HDF5_EXT_ZLIB](#)

6.20.2 Function Documentation

6.20.2.1 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*, [GridObj](#) * *g*, T * *data*, hid_t *hdf_datatype*, bool * *TL_present*, int *TL_thickness*, [HDFstruct](#) *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.

Parameters

<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>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_present</i>	pointer to array of flags indicating whether a lower TL is present on this grid in given direction so offset in file can be computed.
<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.

6.21 HDFstruct.h File Reference

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

Classes

- struct [HDFstruct](#)
Structure for storing halo information for HDF5.

6.22 IBBody.cpp File Reference

```
#include "../inc/stdafx.h"
#include "../inc/IBBody.h"
#include "../inc/IBMarker.h"
#include "../inc/PCpts.h"
#include "../inc/ObjectManager.h"
```

6.23 IBBody.h File Reference

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

Classes

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

6.24 IBInfo.cpp File Reference

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

6.25 IBInfo.h File Reference

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

Classes

- class [IBInfo](#)
Structure for passing IB information between MPI processes.

6.26 IBMarker.cpp File Reference

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

6.27 IBMarker.h File Reference

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

Classes

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

6.28 IVector.h File Reference

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


Classes

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

6.29 main_lbm.cpp File Reference

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

Functions

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

6.29.1 Function Documentation

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

Entry point for the application.

6.30 Marker.h File Reference

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

Classes

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

6.31 MarkerData.h File Reference

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

Classes

- class [MarkerData](#)
Container class to hold marker information.

6.32 `Mpi_buffer_pack.cpp` File Reference

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

6.33 `Mpi_buffer_size_recv.cpp` File Reference

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

6.34 `Mpi_buffer_size_send.cpp` File Reference

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

6.35 `Mpi_buffer_unpk.cpp` File Reference

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

6.36 `MpiManager.cpp` File Reference

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

6.37 `MpiManager.h` File Reference

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

Classes

- class [MpiManager](#)
MPI Manager class.
- struct [MpiManager::layer_edges](#)
Structure containing absolute 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.

6.37.1 Macro Definition Documentation

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

For loop definition for left halo.

6.37.1.2 `#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.

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

For loop definition for bottom halo.

6.37.1.4 `#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.

6.37.1.5 `#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.

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

For loop definition for front halo.

6.38 ObjectManager.cpp File Reference

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

6.39 ObjectManager.h File Reference

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

Classes

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

6.40 ObjectManager_init_bflbody.cpp File Reference

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

6.41 ObjectManager_init_ibmbody.cpp File Reference

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

6.42 ObjectManager_ops_ibm.cpp File Reference

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

6.43 ObjectManager_ops_ibmflex.cpp File Reference

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

Macros

- `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`
Pointer swap definition.
- `#define TINY 1.0e-20`
Definition of small number (could use numerics since this is C++ but nevermind)
- `#define SWAP(a, b) {dum=(a);(a)=(b);(b)=dum;}`
Pointer swap definition.

6.43.1 Macro Definition Documentation

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

Pointer swap definition.

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

Pointer swap definition.

6.43.1.3 `#define TINY 1.0e-20`

Definition of small number (could use numerics since this is C++ but nevermind)

6.44 ObjectManager_ops_io.cpp File Reference

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

6.45 PCpts.h File Reference

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

Classes

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

6.46 stdafx.cpp File Reference

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

Variables

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

6.46.1 Variable Documentation

6.46.1.1 const int [c](#)[3][[L_NUM_VELS](#)]

Initial value:

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

Lattice velocities.

6.46.1.2 const int [c_opt](#)[[L_NUM_VELS](#)][3]

Initial value:

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

Lattice velocities optimised arrangement.

6.46.1.3 `const double cs = 1.0 / sqrt(3.0)`

Lattice sound speed.

6.46.1.4 `const double w[L_NUM_VELS]`

Initial value:

```
=
{ 1.0 / 9.0, 1.0 / 9.0, 1.0 / 9.0, 1.0 / 9.0, 1.0 / 36.0, 1.0 / 36.0, 1.0 / 36.0, 1.0 / 36.0, 4.0 / 9.0 }
```

Quadrature weights.

6.47 stdafx.h File Reference

```
#include <algorithm>
#include <cmath>
#include <vector>
#include <iostream>
#include <fstream>
#include <sstream>
#include <numeric>
#include <assert.h>
#include <stdlib.h>
#include <cstring>
#include <stdio.h>
#include "Enumerations.h"
#include "definitions.h"
#include "GridManager.h"
#include <mpi.h>
#include "MpiManager.h"
#include "GridUtils.h"
```

Macros

- `#define DEPRECATED`
- `#define L_IS_NAN std::isnan`
Not a Number declaration (Unix)
- `#define SQ(x) ((x) * (x))`
- `#define L_SMALL_NUMBER 1e-8`
- `#define LUMA_FAILED 12345`
- `#define L_DACTION_WRITE_OUT_FORCES`
- `#define L_ERROR errorfcn`
Error definition.
- `#define L_INFO infofcn`
Regular writer.

Functions

- void `errorfcn` (const std::string &msg, std::ofstream *logfile)
Fatal Error function.
- void `infofcn` (const std::string &msg, std::ofstream *logfile)
Info / logger function.

Variables

- const int `c` [3][`L_NUM_VELS`]
Lattice velocities.
- const int `c_opt` [`L_NUM_VELS`][3]
Lattice velocities optimised arrangement.
- const double `w` [`L_NUM_VELS`]
Quadrature weights.
- const double `cs`
Lattice sound speed.

6.47.1 Macro Definition Documentation

6.47.1.1 `#define DEPRECATED`

6.47.1.2 `#define L_DACTION_WRITE_OUT_FORCES`

6.47.1.3 `#define L_ERROR errorfcn`

Error definition.

Error function shorthand

6.47.1.4 `#define L_INFO infofcn`

Regular writer.

Info function shorthand

6.47.1.5 `#define L_IS_NAN std::isnan`

Not a Number declaration (Unix)

6.47.1.6 `#define L_SMALL_NUMBER 1e-8`

6.47.1.7 `#define LUMA_FAILED 12345`

6.47.1.8 `#define SQ(x) ((x) * (x))`

6.47.2 Function Documentation

6.47.2.1 `void errorfcn (const std::string & msg, std::ofstream * logfile)` `[inline]`

Fatal Error function.

Writes error to the user and further information to the supplied logfile. Inlined since this header is included everywhere.

Parameters

<i>msg</i>	string to be printed to the log file.
<i>logfile</i>	pointer to the logfile where the message is to be written.

6.47.2.2 `void infofcn (const std::string & msg, std::ofstream * logfile)` `[inline]`

Info / logger function.

Writes string to the supplied logfile. Inlined since this header is included everywhere.

Parameters

<i>msg</i>	string to be printed to the log file.
<i>logfile</i>	pointer to the logfile where the message is to be written.

6.47.3 Variable Documentation

6.47.3.1 `const int c[3][L_NUM_VELS]`

Lattice velocities.

6.47.3.2 `const int c_opt[L_NUM_VELS][3]`

Lattice velocities optimised arrangement.

6.47.3.3 `const double cs`

Lattice sound speed.

6.47.3.4 `const double w[L_NUM_VELS]`

Quadrature weights.

Index

- _Owner
 - Body, [16](#)
 - ~BFLBody
 - BFLBody, [10](#)
 - ~BFLMarker
 - BFLMarker, [12](#)
 - ~Body
 - Body, [14](#)
 - ~GridObj
 - GridObj, [23](#)
 - ~GridUnits
 - GridUnits, [32](#)
 - ~IBBody
 - IBBody, [51](#)
 - ~IBMarker
 - IBMarker, [57](#)
 - ~IVector
 - IVector, [59](#)
 - ~Marker
 - Marker, [63](#)
 - ~MarkerData
 - MarkerData, [65](#)
 - ~PCpts
 - PCpts, [83](#)
- add
 - GridUtils, [35](#)
- addMarker
 - Body, [14](#)
 - IBBody, [52](#)
- BCs
 - IBBody, [54](#)
- BFLBody, [9](#)
 - ~BFLBody, [10](#)
 - BFLBody, [10](#)
 - BFLMarker, [12](#)
 - computeQ, [10](#), [11](#)
 - GridObj, [11](#)
 - Q, [11](#)
- BFLBody.cpp, [85](#)
- BFLBody.h, [85](#)
- BFLMarker, [11](#)
 - ~BFLMarker, [12](#)
 - BFLBody, [12](#)
 - BFLMarker, [12](#)
- BFLMarker.cpp, [85](#)
- BFLMarker.h, [85](#)
- bc_applyBfl
 - GridObj, [23](#)
- bc_applyBounceBack
 - GridObj, [23](#)
- bc_applyExtrapolation
 - GridObj, [24](#)
- bc_applyNrbc
 - GridObj, [24](#)
- bc_applyRegularised
 - GridObj, [24](#)
- bc_applySpecReflect
 - GridObj, [25](#)
- bfl_buildBody
 - ObjectManager, [75](#)
- Body
 - _Owner, [16](#)
 - ~Body, [14](#)
 - addMarker, [14](#)
 - Body, [14](#)
 - buildFromCloud, [15](#)
 - closed_surface, [16](#)
 - getMarkerData, [15](#)
 - id, [16](#)
 - markers, [16](#)
 - passToVoxelFilter, [15](#)
 - spacing, [16](#)
- Body< MarkerType >, [13](#)
- Body.h, [86](#)
- buffer_rcv_info
 - MpiManager, [71](#)
- buffer_send_info
 - MpiManager, [71](#)
- buildFromCloud
 - Body, [15](#)
- c
 - stdafx.cpp, [118](#)
 - stdafx.h, [121](#)
- c_opt
 - stdafx.cpp, [118](#)
 - stdafx.h, [121](#)
- cNumProbes
 - definitions.h, [102](#)
- cProbeLimsX
 - definitions.h, [102](#)
- cProbeLimsY
 - definitions.h, [103](#)
- cProbeLimsZ
 - definitions.h, [103](#)
- cRankSizeX
 - MpiManager, [71](#)
- cRankSizeY

- MpiManager, 71
- cRankSizeZ
 - MpiManager, 71
- cRefEndX
 - definitions.h, 103
- cRefEndY
 - definitions.h, 103
- cRefEndZ
 - definitions.h, 103
- cRefStartX
 - definitions.h, 103
- cRefStartY
 - definitions.h, 103
- cRefStartZ
 - definitions.h, 104
- closed_surface
 - Body, 16
- computeLiftDrag
 - ObjectManager, 76
- computeQ
 - BFLBody, 10, 11
- createOutputDirectory
 - GridUtils, 35
- createWritableDataStore
 - GridManager, 18
- crossprod
 - GridUtils, 35
- cs
 - stdafx.cpp, 118
 - stdafx.h, 121
- DEPRECATED
 - stdafx.h, 120
- definitions.h, 86
 - cNumProbes, 102
 - cProbeLimsX, 102
 - cProbeLimsY, 103
 - cProbeLimsZ, 103
 - cRefEndX, 103
 - cRefEndY, 103
 - cRefEndZ, 103
 - cRefStartX, 103
 - cRefStartY, 103
 - cRefStartZ, 104
 - L_BFL_LENGTH, 91
 - L_BFL_ON_GRID_LEV, 91
 - L_BFL_ON_GRID_REG, 91
 - L_BFL_REF_LENGTH, 91
 - L_BFL_SCALE_DIRECTION, 91
 - L_BLOCK_MAX_X, 92
 - L_BLOCK_MAX_Y, 92
 - L_BLOCK_MAX_Z, 92
 - L_BLOCK_MIN_X, 92
 - L_BLOCK_MIN_Y, 92
 - L_BLOCK_MIN_Z, 92
 - L_BLOCK_ON_GRID_LEV, 92
 - L_BLOCK_ON_GRID_REG, 92
 - L_BUILD_FOR_MPI, 93
 - L_BX, 93
 - L_BY, 93
 - L_BZ, 93
 - L_CENTRE_BFL_Z, 93
 - L_CENTRE_IBB_Z, 93
 - L_CENTRE_OBJECT_Z, 93, 94
 - L_CLOUD_DEBUG, 94
 - L_CSMAG, 94
 - L_DIMS, 94
 - L_FILAMENT_END_BC, 94
 - L_FILAMENT_START_BC, 94
 - L_FREESTREAM_TUNNEL, 94
 - L_GRAVITY_DIRECTION, 94
 - L_GRAVITY_FORCE, 94
 - L_HDF5_OUTPUT, 94
 - L_HDF_DEBUG, 94
 - L_IB_ON_LEV, 95
 - L_IB_ON_REG, 95
 - L_IBB_ANGLE_HORZ, 95
 - L_IBB_ANGLE_VERT, 95
 - L_IBB_DELTA_RHO, 95
 - L_IBB_EI, 95
 - L_IBB_FILAMENT_LENGTH, 95
 - L_IBB_FILAMENT_START_X, 95
 - L_IBB_FILAMENT_START_Y, 96
 - L_IBB_FILAMENT_START_Z, 96
 - L_IBB_FLEXIBLE, 96
 - L_IBB_FROM_FILE, 96
 - L_IBB_LENGTH, 96
 - L_IBB_MOVABLE, 96
 - L_IBB_ON_GRID_LEV, 96
 - L_IBB_ON_GRID_REG, 96
 - L_IBB_REF_LENGTH, 97
 - L_IBB_SCALE_DIRECTION, 97
 - L_IBB_D, 95
 - L_IBB_L, 96
 - L_IBB_R, 96
 - L_IBB_W, 97
 - L_IBB_X, 97
 - L_IBB_Y, 97
 - L_IBB_Z, 97
 - L_INIT_VERBOSE, 97
 - L_INLET_ON, 97
 - L_LD_OUT, 97
 - L_LOG_TIMINGS, 97
 - L_MPI_DIRS, 98
 - L_MPI_VERBOSE, 98
 - L_MPI_WRITE_LOAD_BALANCE, 98
 - L_MPI_XCORES, 98
 - L_MPI_YCORES, 98
 - L_MPI_ZCORES, 98
 - L_NUM_LEVELS, 98
 - L_NUM_MARKERS, 98
 - L_NUM_REGIONS, 98
 - L_NUM_VELS, 99
 - L_OBJECT_LENGTH, 99
 - L_OBJECT_ON_GRID_LEV, 99
 - L_OBJECT_ON_GRID_REG, 99
 - L_OBJECT_REF_LENGTH, 99

- L_OBJECT_SCALE_DIRECTION, 99
- L_OUT_EVERY_FORCES, 99
- L_OUT_EVERY, 99
- L_OUTLET_ON, 99
- L_OUTPUT_PRECISION, 99
- L_PERIODIC_BOUNDARIES, 99
- L_PHYSICAL_U, 100
- L_PROBE_OUT_FREQ, 100
- L_PI, 100
- L_RESOLUTION, 100
- L_RESTART_OUT_FREQ, 100
- L_RHOIN, 100
- L_RE, 100
- L_SOLID_FROM_FILE, 100
- L_START_BFL_X, 100
- L_START_BFL_Y, 100
- L_START_IBB_X, 101
- L_START_IBB_Y, 101
- L_START_OBJECT_X, 101
- L_START_OBJECT_Y, 101
- L_TIMESTEP, 101
- L_TOTAL_TIMESTEPS, 101
- L_UMAX, 101
- L_UREF, 101
- L_UX0, 101
- L_UY0, 101
- L_UZ0, 102
- L_VTK_BODY_WRITE, 102
- L_WALL_THICKNESS_BACK, 102
- L_WALL_THICKNESS_BOTTOM, 102
- L_WALL_THICKNESS_FRONT, 102
- L_WALL_THICKNESS_TOP, 102
- L_K, 97
- L_M, 98
- L_N, 98
- LUMA_VERSION, 102
- delta_rho
 - IBBody, 54
- deltaval
 - IBMarker, 58
- desired_vel
 - IBMarker, 58
- destroyInstance
 - GridManager, 19
 - MpiManager, 68
 - ObjectManager, 76
- dh
 - GridObj, 29
- dilation
 - IBMarker, 58
- dimensions
 - MpiManager, 71
- dir_reflect
 - GridUtils, 48
- dotprod
 - GridUtils, 36
- downToLimit
 - GridUtils, 36
- dt
 - GridObj, 29
- eBBBCloud
 - Enumerations.h, 107
- eBCAll
 - Enumerations.h, 105
- eBCBFL
 - Enumerations.h, 105
- eBCInlet
 - Enumerations.h, 105
- eBCInletOutlet
 - Enumerations.h, 105
- eBCOutlet
 - Enumerations.h, 105
- eBCSolidSymmetry
 - Enumerations.h, 105
- eBCType
 - Enumerations.h, 105
- eBFLCloud
 - Enumerations.h, 107
- eBFL
 - Enumerations.h, 107
- eCartMinMax
 - Enumerations.h, 105
- eCartesianDirection
 - Enumerations.h, 105
- eCore
 - Enumerations.h, 106
- eEdgeMinMax
 - Enumerations.h, 105
- eFluid
 - Enumerations.h, 107
- eHalo
 - Enumerations.h, 106
- eHdf5SlabType
 - Enumerations.h, 105
- eIBBCloud
 - Enumerations.h, 107
- eIBDeltaSum
 - Enumerations.h, 106
- eIBEpsilon
 - Enumerations.h, 106
- eIBInfoType
 - Enumerations.h, 106
- eIBMarkerPositions
 - Enumerations.h, 106
- eIBVelocityInterpolation
 - Enumerations.h, 106
- eIBVelocitySpreading
 - Enumerations.h, 106
- eOFlag
 - Enumerations.h, 106
- eInlet
 - Enumerations.h, 107
- eLeftMax
 - Enumerations.h, 105
- eLeftMin
 - Enumerations.h, 105

- eLocationOnRank
 - Enumerations.h, [106](#)
- eMaximum
 - Enumerations.h, [107](#)
- eMinMax
 - Enumerations.h, [106](#)
- eMinimum
 - Enumerations.h, [107](#)
- eNone
 - Enumerations.h, [106](#)
- eObjectType
 - Enumerations.h, [107](#)
- eOutlet
 - Enumerations.h, [107](#)
- ePosX
 - Enumerations.h, [106](#)
- ePosY
 - Enumerations.h, [106](#)
- ePosZ
 - Enumerations.h, [106](#)
- eProductVector
 - Enumerations.h, [106](#)
- eRead
 - Enumerations.h, [106](#)
- eRefined
 - Enumerations.h, [107](#)
- eRefinedInlet
 - Enumerations.h, [107](#)
- eRefinedSolid
 - Enumerations.h, [107](#)
- eRefinedSymmetry
 - Enumerations.h, [107](#)
- eRightMax
 - Enumerations.h, [105](#)
- eRightMin
 - Enumerations.h, [105](#)
- eScalar
 - Enumerations.h, [106](#)
- eSolid
 - Enumerations.h, [107](#)
- eSymmetry
 - Enumerations.h, [107](#)
- eTransitionToCoarser
 - Enumerations.h, [107](#)
- eTransitionToFiner
 - Enumerations.h, [107](#)
- eType
 - Enumerations.h, [107](#)
- eVector
 - Enumerations.h, [106](#)
- eWrite
 - Enumerations.h, [106](#)
- eXDirection
 - Enumerations.h, [105](#)
- eXMax
 - Enumerations.h, [105](#)
- eXMin
 - Enumerations.h, [105](#)
- eYDirection
 - Enumerations.h, [105](#)
- eYMax
 - Enumerations.h, [105](#)
- eYMin
 - Enumerations.h, [105](#)
- eZDirection
 - Enumerations.h, [105](#)
- eZMax
 - Enumerations.h, [105](#)
- eZMin
 - Enumerations.h, [105](#)
- Enumerations.h, [104](#)
 - eBBBCloud, [107](#)
 - eBCAll, [105](#)
 - eBCBFL, [105](#)
 - eBCInlet, [105](#)
 - eBCInletOutlet, [105](#)
 - eBCOutlet, [105](#)
 - eBCSolidSymmetry, [105](#)
 - eBCType, [105](#)
 - eBFLCloud, [107](#)
 - eBFL, [107](#)
 - eCartMinMax, [105](#)
 - eCartesianDirection, [105](#)
 - eCore, [106](#)
 - eEdgeMinMax, [105](#)
 - eFluid, [107](#)
 - eHalo, [106](#)
 - eHdf5SlabType, [105](#)
 - eIBBCloud, [107](#)
 - eIBDeltaSum, [106](#)
 - eIBEpsilon, [106](#)
 - eIBInfoType, [106](#)
 - eIBMarkerPositions, [106](#)
 - eIBVelocityInterpolation, [106](#)
 - eIBVelocitySpreading, [106](#)
 - eIOFlag, [106](#)
 - eInlet, [107](#)
 - eLeftMax, [105](#)
 - eLeftMin, [105](#)
 - eLocationOnRank, [106](#)
 - eMaximum, [107](#)
 - eMinMax, [106](#)
 - eMinimum, [107](#)
 - eNone, [106](#)
 - eObjectType, [107](#)
 - eOutlet, [107](#)
 - ePosX, [106](#)
 - ePosY, [106](#)
 - ePosZ, [106](#)
 - eProductVector, [106](#)
 - eRead, [106](#)
 - eRefined, [107](#)
 - eRefinedInlet, [107](#)
 - eRefinedSolid, [107](#)
 - eRefinedSymmetry, [107](#)
 - eRightMax, [105](#)

- eRightMin, [105](#)
- eScalar, [106](#)
- eSolid, [107](#)
- eSymmetry, [107](#)
- eTransitionToCoarser, [107](#)
- eTransitionToFiner, [107](#)
- eType, [107](#)
- eVector, [106](#)
- eWrite, [106](#)
- eXDirection, [105](#)
- eXMax, [105](#)
- eXMin, [105](#)
- eYDirection, [105](#)
- eYMax, [105](#)
- eYMin, [105](#)
- eZDirection, [105](#)
- eZMax, [105](#)
- eZMin, [105](#)
- epsilon
 - IBMarker, [58](#)
- errorfcn
 - stdafx.h, [120](#)
- f_buffer_recv
 - MpiManager, [71](#)
- f_buffer_send
 - MpiManager, [71](#)
- factorial
 - GridUtils, [36](#)
- flexural_rigidity
 - IBBody, [54](#)
- fluid_vel
 - IBMarker, [58](#)
- force_xyz
 - IBMarker, [58](#)
- getCoarseIndices
 - GridUtils, [37](#)
- getEnclosingVoxel
 - GridUtils, [37](#)
- getFineIndices
 - GridUtils, [38](#)
- getGrid
 - GridUtils, [38](#)
- getInstance
 - GridManager, [19](#)
 - MpiManager, [68](#)
 - ObjectManager, [76](#)
- getMarkerData
 - Body, [15](#)
- getMpiDirection
 - GridUtils, [39](#)
- getOpposite
 - GridUtils, [39](#)
- global_edges
 - GridManager, [19](#)
- global_size
 - GridManager, [19](#)
- GridManager, [17](#)
- createWritableDataStore, [18](#)
- destroyInstance, [19](#)
- getInstance, [19](#)
- global_edges, [19](#)
- global_size, [19](#)
- GridObj, [19](#)
- GridUtils, [19](#)
- Grids, [19](#)
- local_size, [20](#)
- MpiManager, [19](#)
- p_data, [20](#)
- setLocalCoarseSize, [19](#)
- subgrid_tlayer_key, [20](#)
- GridManager.cpp, [107](#)
- GridManager.h, [108](#)
- GridObj, [20](#)
 - ~GridObj, [23](#)
 - BFLBody, [11](#)
 - bc_applyBfl, [23](#)
 - bc_applyBounceBack, [23](#)
 - bc_applyExtrapolation, [24](#)
 - bc_applyNrbc, [24](#)
 - bc_applyRegularised, [24](#)
 - bc_applySpecReflect, [25](#)
 - dh, [29](#)
 - dt, [29](#)
 - GridManager, [19](#)
 - GridObj, [23](#)
 - GridUtils, [29](#)
 - io_fgout, [25](#)
 - io_hdf5, [25](#)
 - io_lite, [25](#)
 - io_probeOutput, [25](#)
 - io_restart, [26](#)
 - io_textout, [26](#)
 - K_lim, [29](#)
 - LBM_addSubGrid, [26](#)
 - LBM_init_getInletProfile, [26](#)
 - LBM_initBoundLab, [26](#)
 - LBM_initGrid, [27](#)
 - LBM_initGridToGridMappings, [27](#)
 - LBM_initPositionVector, [27](#)
 - LBM_initRefinedLab, [27](#)
 - LBM_initRho, [27](#)
 - LBM_initSolidLab, [28](#)
 - LBM_initSubGrid, [28](#)
 - LBM_initVelocity, [28](#)
 - LBM_kbcCollide, [28](#)
 - LBM_macro, [28](#)
 - LBM_multi_opt, [29](#)
 - LBM_resetForces, [29](#)
 - LatTyp, [29](#)
 - level, [30](#)
 - M_lim, [30](#)
 - MpiManager, [29](#)
 - N_lim, [30](#)
 - nu, [30](#)
 - ObjectManager, [29, 82](#)

- omega, 30
- region_number, 30
- t, 30
- timeav_mpi_overhead, 30
- timeav_timestep, 30
- XOrigin, 30
- XPos, 31
- YOrigin, 31
- YPos, 31
- ZOrigin, 31
- ZPos, 31
- GridObj.cpp, 108
- GridObj.h, 108
- GridObj_init_grids.cpp, 108
- GridObj_ops_boundary.cpp, 108
- GridObj_ops_io.cpp, 109
- GridObj_ops_lbm.cpp, 109
- GridObj_ops_lbm_optimised.cpp, 109
- GridUnits, 31
 - ~GridUnits, 32
 - GridUnits, 32
 - m2cm, 32
 - ulat2uphys, 32
- GridUnits.h, 109
- GridUtils, 33
 - add, 35
 - createOutputDirectory, 35
 - crossprod, 35
 - dir_reflect, 48
 - dotprod, 36
 - downToLimit, 36
 - factorial, 36
 - getCoarseIndices, 37
 - getEnclosingVoxel, 37
 - getFineIndices, 38
 - getGrid, 38
 - getMpiDirection, 39
 - getOpposite, 39
 - GridManager, 19
 - GridObj, 29
 - intersectsRefinedRegion, 39
 - isOffGrid, 40
 - isOnRecvLayer, 40
 - isOnSenderLayer, 41
 - isOnThisRank, 41, 42
 - isOnTransitionLayer, 42, 43
 - isOverlapPeriodic, 43
 - linspace, 43
 - logfile, 48
 - matrix_multiply, 44
 - onespace, 44
 - path_str, 48
 - safeGetRank, 44
 - stridedCopy, 44
 - subtract, 45
 - upToZero, 45
 - vecmultiply, 45
 - vecnorm, 46, 47
 - GridUtils.cpp, 109
 - GridUtils.h, 109
 - Grids
 - GridManager, 19
 - groupID
 - IBBody, 54
 - H5_BUILT_AS_DYNAMIC_LIB
 - hdf5luma.h, 110
 - HDF5_EXT_SZIP
 - hdf5luma.h, 110
 - HDF5_EXT_ZLIB
 - hdf5luma.h, 110
 - HDFstruct, 48
 - i_end, 49
 - i_start, 49
 - j_end, 49
 - j_start, 49
 - k_end, 49
 - k_start, 49
 - level, 49
 - region, 49
 - writable_data_count, 49
 - HDFstruct.h, 111
 - hdf5_writeDataSet
 - hdf5luma.h, 110
 - hdf5luma.h, 110
 - H5_BUILT_AS_DYNAMIC_LIB, 110
 - HDF5_EXT_SZIP, 110
 - HDF5_EXT_ZLIB, 110
 - hdf5_writeDataSet, 110
 - i
 - MarkerData, 65
 - i_end
 - HDFstruct, 49
 - i_start
 - HDFstruct, 49
 - IBBody, 50
 - ~IBBody, 51
 - addMarker, 52
 - BCs, 54
 - delta_rho, 54
 - flexural_rigidity, 54
 - groupID, 54
 - IBBody, 51, 52
 - IBInfo, 54
 - IBMarker, 57
 - isFlexible, 54
 - isMovable, 54
 - makeBody, 52, 53
 - ObjectManager, 54
 - tension, 54
 - IBBody.cpp, 111
 - IBBody.h, 111
 - IBInfo, 55
 - IBBody, 54
 - IBInfo, 55
 - IBMarker, 57

- mapToMpiStruct, 55
- IBInfo.cpp, 112
- IBInfo.h, 112
- IBMarker, 56
 - ~IBMarker, 57
 - deltaval, 58
 - desired_vel, 58
 - dilation, 58
 - epsilon, 58
 - fluid_vel, 58
 - force_xyz, 58
 - IBBody, 57
 - IBInfo, 57
 - IBMarker, 57
 - isFlexible, 58
 - local_area, 58
 - ObjectManager, 57
 - position_old, 58
- IBMarker.cpp, 112
- IBMarker.h, 112
- IVector
 - ~IVector, 59
 - IVector, 59
 - operator(), 60
- IVector< GenTyp >, 59
- IVector.h, 112
- ibm_apply
 - ObjectManager, 76
- ibm_banbks
 - ObjectManager, 76
- ibm_bandec
 - ObjectManager, 77
- ibm_bicgstab
 - ObjectManager, 77
- ibm_buildBody
 - ObjectManager, 78
- ibm_computeForce
 - ObjectManager, 78
- ibm_deltaKernel
 - ObjectManager, 78
- ibm_findEpsilon
 - ObjectManager, 79
- ibm_findSupport
 - ObjectManager, 79
- ibm_initialise
 - ObjectManager, 79
- ibm_initialiseSupport
 - ObjectManager, 79
- ibm_interpol
 - ObjectManager, 80
- ibm_jacowire
 - ObjectManager, 80
- ibm_moveBodies
 - ObjectManager, 80
- ibm_positionUpdate
 - ObjectManager, 80
- ibm_positionUpdateGroup
 - ObjectManager, 80
- ibm_spread
 - ObjectManager, 81
- ID
 - MarkerData, 65
- id
 - Body, 16
- infofcn
 - stdafx.h, 121
- intersectsRefinedRegion
 - GridUtils, 39
- io_fgaout
 - GridObj, 25
- io_hdf5
 - GridObj, 25
- io_lite
 - GridObj, 25
- io_probeOutput
 - GridObj, 25
- io_readInCloud
 - ObjectManager, 81
- io_restart
 - GridObj, 26
 - ObjectManager, 81
- io_textout
 - GridObj, 26
- io_vtkIBBWriter
 - ObjectManager, 81
- io_writeBodyPosition
 - ObjectManager, 81
- io_writeForceOnObject
 - ObjectManager, 82
- io_writeLiftDrag
 - ObjectManager, 82
- isFlexible
 - IBBody, 54
 - IBMarker, 58
- isMovable
 - IBBody, 54
- isOffGrid
 - GridUtils, 40
- isOnRecvLayer
 - GridUtils, 40
- isOnSenderLayer
 - GridUtils, 41
- isOnThisRank
 - GridUtils, 41, 42
- isOnTransitionLayer
 - GridUtils, 42, 43
- isOverlapPeriodic
 - GridUtils, 43
- j
 - MarkerData, 65
- j_end
 - HDFstruct, 49
- j_start
 - HDFstruct, 49
- k

- MarkerData, 65
- k_end
 - HDFstruct, 49
- K_lim
 - GridObj, 29
- k_start
 - HDFstruct, 49
- L_BFL_LENGTH
 - definitions.h, 91
- L_BFL_ON_GRID_LEV
 - definitions.h, 91
- L_BFL_ON_GRID_REG
 - definitions.h, 91
- L_BFL_REF_LENGTH
 - definitions.h, 91
- L_BFL_SCALE_DIRECTION
 - definitions.h, 91
- L_BLOCK_MAX_X
 - definitions.h, 92
- L_BLOCK_MAX_Y
 - definitions.h, 92
- L_BLOCK_MAX_Z
 - definitions.h, 92
- L_BLOCK_MIN_X
 - definitions.h, 92
- L_BLOCK_MIN_Y
 - definitions.h, 92
- L_BLOCK_MIN_Z
 - definitions.h, 92
- L_BLOCK_ON_GRID_LEV
 - definitions.h, 92
- L_BLOCK_ON_GRID_REG
 - definitions.h, 92
- L_BUILD_FOR_MPI
 - definitions.h, 93
- L_BX
 - definitions.h, 93
- L_BY
 - definitions.h, 93
- L_BZ
 - definitions.h, 93
- L_CENTRE_BFL_Z
 - definitions.h, 93
- L_CENTRE_IBB_Z
 - definitions.h, 93
- L_CENTRE_OBJECT_Z
 - definitions.h, 93, 94
- L_CLOUD_DEBUG
 - definitions.h, 94
- L_CSMAG
 - definitions.h, 94
- L_DACTION_WRITE_OUT_FORCES
 - stdafx.h, 120
- L_DIMS
 - definitions.h, 94
- L_ERROR
 - stdafx.h, 120
- L_FILAMENT_END_BC
 - definitions.h, 94
- L_FILAMENT_START_BC
 - definitions.h, 94
- L_FREESTREAM_TUNNEL
 - definitions.h, 94
- L_GRAVITY_DIRECTION
 - definitions.h, 94
- L_GRAVITY_FORCE
 - definitions.h, 94
- L_HDF5_OUTPUT
 - definitions.h, 94
- L_HDF_DEBUG
 - definitions.h, 94
- L_IB_ON_LEV
 - definitions.h, 95
- L_IB_ON_REG
 - definitions.h, 95
- L_IBB_ANGLE_HORZ
 - definitions.h, 95
- L_IBB_ANGLE_VERT
 - definitions.h, 95
- L_IBB_DELTA_RHO
 - definitions.h, 95
- L_IBB_EI
 - definitions.h, 95
- L_IBB_FILAMENT_LENGTH
 - definitions.h, 95
- L_IBB_FILAMENT_START_X
 - definitions.h, 95
- L_IBB_FILAMENT_START_Y
 - definitions.h, 96
- L_IBB_FILAMENT_START_Z
 - definitions.h, 96
- L_IBB_FLEXIBLE
 - definitions.h, 96
- L_IBB_FROM_FILE
 - definitions.h, 96
- L_IBB_LENGTH
 - definitions.h, 96
- L_IBB_MOVABLE
 - definitions.h, 96
- L_IBB_ON_GRID_LEV
 - definitions.h, 96
- L_IBB_ON_GRID_REG
 - definitions.h, 96
- L_IBB_REF_LENGTH
 - definitions.h, 97
- L_IBB_SCALE_DIRECTION
 - definitions.h, 97
- L_IBB_D
 - definitions.h, 95
- L_IBB_L
 - definitions.h, 96
- L_IBB_R
 - definitions.h, 96
- L_IBB_W
 - definitions.h, 97
- L_IBB_X

- definitions.h, 97
- L_IBB_Y
 - definitions.h, 97
- L_IBB_Z
 - definitions.h, 97
- L_INFO
 - stdafx.h, 120
- L_INIT_VERBOSE
 - definitions.h, 97
- L_INLET_ON
 - definitions.h, 97
- L_IS_NAN
 - stdafx.h, 120
- L_LD_OUT
 - definitions.h, 97
- L_LOG_TIMINGS
 - definitions.h, 97
- L_MPI_DIRS
 - definitions.h, 98
- L_MPI_VERBOSE
 - definitions.h, 98
- L_MPI_WRITE_LOAD_BALANCE
 - definitions.h, 98
- L_MPI_XCORES
 - definitions.h, 98
- L_MPI_YCORES
 - definitions.h, 98
- L_MPI_ZCORES
 - definitions.h, 98
- L_NUM_LEVELS
 - definitions.h, 98
- L_NUM_MARKERS
 - definitions.h, 98
- L_NUM_REGIONS
 - definitions.h, 98
- L_NUM_VELS
 - definitions.h, 99
- L_OBJECT_LENGTH
 - definitions.h, 99
- L_OBJECT_ON_GRID_LEV
 - definitions.h, 99
- L_OBJECT_ON_GRID_REG
 - definitions.h, 99
- L_OBJECT_REF_LENGTH
 - definitions.h, 99
- L_OBJECT_SCALE_DIRECTION
 - definitions.h, 99
- L_OUT_EVERY_FORCES
 - definitions.h, 99
- L_OUT_EVERY
 - definitions.h, 99
- L_OUTLET_ON
 - definitions.h, 99
- L_OUTPUT_PRECISION
 - definitions.h, 99
- L_PERIODIC_BOUNDARIES
 - definitions.h, 99
- L_PHYSICAL_U
 - definitions.h, 100
- L_PROBE_OUT_FREQ
 - definitions.h, 100
- L_PI
 - definitions.h, 100
- L_RESOLUTION
 - definitions.h, 100
- L_RESTART_OUT_FREQ
 - definitions.h, 100
- L_RHOIN
 - definitions.h, 100
- L_RE
 - definitions.h, 100
- L_SMALL_NUMBER
 - stdafx.h, 120
- L_SOLID_FROM_FILE
 - definitions.h, 100
- L_START_BFL_X
 - definitions.h, 100
- L_START_BFL_Y
 - definitions.h, 100
- L_START_IBB_X
 - definitions.h, 101
- L_START_IBB_Y
 - definitions.h, 101
- L_START_OBJECT_X
 - definitions.h, 101
- L_START_OBJECT_Y
 - definitions.h, 101
- L_TIMESTEP
 - definitions.h, 101
- L_TOTAL_TIMESTEPS
 - definitions.h, 101
- L_UMAX
 - definitions.h, 101
- L_UREF
 - definitions.h, 101
- L_UX0
 - definitions.h, 101
- L_UY0
 - definitions.h, 101
- L_UZ0
 - definitions.h, 102
- L_VTK_BODY_WRITE
 - definitions.h, 102
- L_WALL_THICKNESS_BACK
 - definitions.h, 102
- L_WALL_THICKNESS_BOTTOM
 - definitions.h, 102
- L_WALL_THICKNESS_FRONT
 - definitions.h, 102
- L_WALL_THICKNESS_TOP
 - definitions.h, 102
- L_K
 - definitions.h, 97
- L_M
 - definitions.h, 98
- L_N

- definitions.h, 98
- LBM_addSubGrid
 - GridObj, 26
- LBM_init_getInletProfile
 - GridObj, 26
- LBM_initBoundLab
 - GridObj, 26
- LBM_initGrid
 - GridObj, 27
- LBM_initGridToGridMappings
 - GridObj, 27
- LBM_initPositionVector
 - GridObj, 27
- LBM_initRefinedLab
 - GridObj, 27
- LBM_initRho
 - GridObj, 27
- LBM_initSolidLab
 - GridObj, 28
- LBM_initSubGrid
 - GridObj, 28
- LBM_initVelocity
 - GridObj, 28
- LBM_kbcCollide
 - GridObj, 28
- LBM_macro
 - GridObj, 28
- LBM_multi_opt
 - GridObj, 29
- LBM_resetForces
 - GridObj, 29
- LUMA_FAILED
 - stdafx.h, 120
- LUMA_VERSION
 - definitions.h, 102
- LatTyp
 - GridObj, 29
- level
 - GridObj, 30
 - HDFstruct, 49
 - MpiManager::buffer_struct, 17
- linspace
 - GridUtils, 43
- local_area
 - IBMarker, 58
- local_size
 - GridManager, 20
- logfile
 - GridUtils, 48
- logout
 - MpiManager, 71
- m2cm
 - GridUnits, 32
- M_lim
 - GridObj, 30
- main
 - main_lbm.cpp, 113
- main_lbm.cpp, 113
 - main, 113
- makeBody
 - IBBody, 52, 53
- mapToMpiStruct
 - IBInfo, 55
- Marker, 62
 - ~Marker, 63
 - Marker, 63
 - position, 63
 - supp_i, 63
 - supp_j, 63
 - supp_k, 63
 - support_rank, 64
- Marker.h, 113
- MarkerData, 64
 - ~MarkerData, 65
 - i, 65
 - ID, 65
 - j, 65
 - k, 65
 - MarkerData, 64, 65
 - x, 65
 - y, 65
 - z, 66
- MarkerData.h, 113
- markers
 - Body, 16
- matrix_multiply
 - GridUtils, 44
- mpi_buffer_pack
 - MpiManager, 68
- Mpi_buffer_pack.cpp, 114
- mpi_buffer_size
 - MpiManager, 68
- mpi_buffer_size_recv
 - MpiManager, 68
- Mpi_buffer_size_recv.cpp, 114
- mpi_buffer_size_send
 - MpiManager, 69
- Mpi_buffer_size_send.cpp, 114
- mpi_buffer_unpack
 - MpiManager, 69
- Mpi_buffer_unpk.cpp, 114
- mpi_buildCommunicators
 - MpiManager, 69
- mpi_communicate
 - MpiManager, 69
- mpi_getOpposite
 - MpiManager, 70
- mpi_gridbuild
 - MpiManager, 70
- mpi_init
 - MpiManager, 70
- mpi_updateLoadInfo
 - MpiManager, 70
- mpi_writeout_buf
 - MpiManager, 71
- MpiManager, 66

- buffer_recv_info, [71](#)
- buffer_send_info, [71](#)
- cRankSizeX, [71](#)
- cRankSizeY, [71](#)
- cRankSizeZ, [71](#)
- destroyInstance, [68](#)
- dimensions, [71](#)
- f_buffer_recv, [71](#)
- f_buffer_send, [71](#)
- getInstance, [68](#)
- GridManager, [19](#)
- GridObj, [29](#)
- logout, [71](#)
- mpi_buffer_pack, [68](#)
- mpi_buffer_size, [68](#)
- mpi_buffer_size_recv, [68](#)
- mpi_buffer_size_send, [69](#)
- mpi_buffer_unpack, [69](#)
- mpi_buildCommunicators, [69](#)
- mpi_communicate, [69](#)
- mpi_getOpposite, [70](#)
- mpi_gridbuild, [70](#)
- mpi_init, [70](#)
- mpi_updateLoadInfo, [70](#)
- mpi_writeout_buf, [71](#)
- my_rank, [72](#)
- neighbour_coords, [72](#)
- neighbour_rank, [72](#)
- neighbour_vectors, [72](#)
- num_ranks, [72](#)
- rank_coords, [72](#)
- rank_core_edge, [72](#)
- recv_layer_pos, [72](#)
- recv_stat, [73](#)
- send_requests, [73](#)
- send_stat, [73](#)
- sender_layer_pos, [73](#)
- subGrid_comm, [73](#)
- world_comm, [73](#)
- MpiManager.cpp, [114](#)
- MpiManager.h, [114](#)
 - range_i_left, [115](#)
 - range_i_right, [115](#)
 - range_j_down, [115](#)
 - range_j_up, [115](#)
 - range_k_back, [115](#)
 - range_k_front, [115](#)
- MpiManager::buffer_struct, [16](#)
 - level, [17](#)
 - region, [17](#)
 - size, [17](#)
- MpiManager::layer_edges, [61](#)
 - X, [61](#)
 - Y, [61](#)
 - Z, [62](#)
- my_rank
 - MpiManager, [72](#)
- N_lim
 - GridObj, [30](#)
- neighbour_coords
 - MpiManager, [72](#)
- neighbour_rank
 - MpiManager, [72](#)
- neighbour_vectors
 - MpiManager, [72](#)
- nu
 - GridObj, [30](#)
- num_ranks
 - MpiManager, [72](#)
- ObjectManager, [73](#)
 - bfl_buildBody, [75](#)
 - computeLiftDrag, [76](#)
 - destroyInstance, [76](#)
 - getInstance, [76](#)
 - GridObj, [29](#), [82](#)
 - IBBody, [54](#)
 - IBMarker, [57](#)
 - ibm_apply, [76](#)
 - ibm_banbks, [76](#)
 - ibm_bandec, [77](#)
 - ibm_bicgstab, [77](#)
 - ibm_buildBody, [78](#)
 - ibm_computeForce, [78](#)
 - ibm_deltaKernel, [78](#)
 - ibm_findEpsilon, [79](#)
 - ibm_findSupport, [79](#)
 - ibm_initialise, [79](#)
 - ibm_initialiseSupport, [79](#)
 - ibm_interpol, [80](#)
 - ibm_jacowire, [80](#)
 - ibm_moveBodies, [80](#)
 - ibm_positionUpdate, [80](#)
 - ibm_positionUpdateGroup, [80](#)
 - ibm_spread, [81](#)
 - io_readInCloud, [81](#)
 - io_restart, [81](#)
 - io_vtkIBBWriter, [81](#)
 - io_writeBodyPosition, [81](#)
 - io_writeForceOnObject, [82](#)
 - io_writeLiftDrag, [82](#)
- ObjectManager.cpp, [116](#)
- ObjectManager.h, [116](#)
- ObjectManager_init_bflbody.cpp, [116](#)
- ObjectManager_init_ibmbody.cpp, [116](#)
- ObjectManager_ops_ibm.cpp, [116](#)
- ObjectManager_ops_ibmflex.cpp, [117](#)
 - SWAP, [117](#)
 - TINY, [117](#)
- ObjectManager_ops_io.cpp, [117](#)
- omega
 - GridObj, [30](#)
- onespace
 - GridUtils, [44](#)
- operator()
 - IVector, [60](#)

- p_data
 - GridManager, 20
- PCpts, 82
 - ~PCpts, 83
 - PCpts, 83
 - x, 83
 - y, 83
 - z, 83
- PCpts.h, 117
- passToVoxelFilter
 - Body, 15
- path_str
 - GridUtils, 48
- position
 - Marker, 63
- position_old
 - IBMarker, 58
- Q
 - BFLBody, 11
- range_i_left
 - MpiManager.h, 115
- range_i_right
 - MpiManager.h, 115
- range_j_down
 - MpiManager.h, 115
- range_j_up
 - MpiManager.h, 115
- range_k_back
 - MpiManager.h, 115
- range_k_front
 - MpiManager.h, 115
- rank_coords
 - MpiManager, 72
- rank_core_edge
 - MpiManager, 72
- recv_layer_pos
 - MpiManager, 72
- recv_stat
 - MpiManager, 73
- region
 - HDFstruct, 49
 - MpiManager::buffer_struct, 17
- region_number
 - GridObj, 30
- SWAP
 - ObjectManager_ops_ibmflex.cpp, 117
- safeGetRank
 - GridUtils, 44
- send_requests
 - MpiManager, 73
- send_stat
 - MpiManager, 73
- sender_layer_pos
 - MpiManager, 73
- setLocalCoarseSize
 - GridManager, 19
- size
 - MpiManager::buffer_struct, 17
- spacing
 - Body, 16
- SQ
 - stdafx.h, 120
- stdafx.cpp, 118
 - c, 118
 - c_opt, 118
 - cs, 118
 - w, 119
- stdafx.h, 119
 - c, 121
 - c_opt, 121
 - cs, 121
 - DEPRECATED, 120
 - errorfcn, 120
 - infofcn, 121
 - L_DACTION_WRITE_OUT_FORCES, 120
 - L_ERROR, 120
 - L_INFO, 120
 - L_IS_NAN, 120
 - L_SMALL_NUMBER, 120
 - LUMA_FAILED, 120
 - SQ, 120
 - w, 121
- stridedCopy
 - GridUtils, 44
- subGrid_comm
 - MpiManager, 73
- subgrid_tlayer_key
 - GridManager, 20
- subtract
 - GridUtils, 45
- supp_i
 - Marker, 63
- supp_j
 - Marker, 63
- supp_k
 - Marker, 63
- support_rank
 - Marker, 64
- t
 - GridObj, 30
- TINY
 - ObjectManager_ops_ibmflex.cpp, 117
- tension
 - IBBody, 54
- timeav_mpi_overhead
 - GridObj, 30
- timeav_timestep
 - GridObj, 30
- ulat2uphys
 - GridUnits, 32
- upToZero
 - GridUtils, 45

- vecmultiply
 - GridUtils, [45](#)
- vecnorm
 - GridUtils, [46](#), [47](#)
- w
 - stdafx.cpp, [119](#)
 - stdafx.h, [121](#)
- world_comm
 - MpiManager, [73](#)
- writable_data_count
 - HDFstruct, [49](#)
- X
 - MpiManager::layer_edges, [61](#)
- x
 - MarkerData, [65](#)
 - PCpts, [83](#)
- XOrigin
 - GridObj, [30](#)
- XPos
 - GridObj, [31](#)
- Y
 - MpiManager::layer_edges, [61](#)
- y
 - MarkerData, [65](#)
 - PCpts, [83](#)
- YOrigin
 - GridObj, [31](#)
- YPos
 - GridObj, [31](#)
- Z
 - MpiManager::layer_edges, [62](#)
- z
 - MarkerData, [66](#)
 - PCpts, [83](#)
- ZOrigin
 - GridObj, [31](#)
- ZPos
 - GridObj, [31](#)