

LUMA

1.2.0-alpha

Generated by Doxygen 1.8.11

Contents

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

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

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

5.5.3.21	LBM_initBoundLab()	25
5.5.3.22	LBM_initGrid()	26
5.5.3.23	LBM_initGrid(std::vector< int > local_size, std::vector< std::vector< int > > GlobalLimsInd, std::vector< std::vector< double > > GlobalLimsPos)	26
5.5.3.24	LBM_initRefinedLab(GridObj &pGrid)	26
5.5.3.25	LBM_initRho()	26
5.5.3.26	LBM_initSolidLab()	26
5.5.3.27	LBM_initSubGrid(GridObj &pGrid)	26
5.5.3.28	LBM_initVelocity()	27
5.5.3.29	LBM_kbcCollide(int i, int j, int k, IVector< double > &f_new)	27
5.5.3.30	LBM_macro()	27
5.5.3.31	LBM_macro(int i, int j, int k)	27
5.5.3.32	LBM_multi(bool ibmFlag)	27
5.5.3.33	LBM_resetForces()	28
5.5.3.34	LBM_stream()	28
5.5.4	Friends And Related Function Documentation	28
5.5.4.1	GridUtils	28
5.5.4.2	MpiManager	28
5.5.4.3	ObjectManager	28
5.5.5	Member Data Documentation	28
5.5.5.1	dt	28
5.5.5.2	K_lim	28
5.5.5.3	LatTyp	28
5.5.5.4	level	28
5.5.5.5	M_lim	29
5.5.5.6	N_lim	29
5.5.5.7	nu	29
5.5.5.8	omega	29
5.5.5.9	t	29
5.5.5.10	timeav_mpi_overhead	29
5.5.5.11	timeav_timestep	29

5.5.5.12	XInd	29
5.5.5.13	XPos	29
5.5.5.14	YInd	29
5.5.5.15	YPos	30
5.5.5.16	ZInd	30
5.5.5.17	ZPos	30
5.6	GridUtils Class Reference	30
5.6.1	Detailed Description	32
5.6.2	Member Function Documentation	32
5.6.2.1	add(std::vector< double > a, std::vector< double > b)	32
5.6.2.2	createOutputDirectory(std::string path_str)	33
5.6.2.3	crossprod(std::vector< double > vec1, std::vector< double > vec2)	33
5.6.2.4	dotprod(std::vector< double > vec1, std::vector< double > vec2)	33
5.6.2.5	downToLimit(NumType x, NumType limit)	33
5.6.2.6	factorial(NumType n)	34
5.6.2.7	getCoarseIndices(int fine_i, int x_start, int fine_j, int y_start, int fine_k, int z_start)	34
5.6.2.8	getFineIndices(int coarse_i, int x_start, int coarse_j, int y_start, int coarse_k, int z_start)	35
5.6.2.9	getGrid(GridObj *&Grids, int level, int region, GridObj *&ptr)	35
5.6.2.10	getOpposite(int direction)	35
5.6.2.11	getVoxInd(double x, double y, double z)	36
5.6.2.12	getVoxInd(double p)	36
5.6.2.13	global_to_local(int i, int j, int k, GridObj *g, std::vector< NumType > &locals)	36
5.6.2.14	hasThisSubGrid(const GridObj &pGrid, int RegNum)	37
5.6.2.15	isOffGrid(int i, int j, int k, GridObj &g)	37
5.6.2.16	isOnRecvLayer(double pos_x, double pos_y, double pos_z)	37
5.6.2.17	isOnRecvLayer(double site_position, enum eCartesianDirection xyz, enum e↔MinMax minmax)	38
5.6.2.18	isOnSenderLayer(double pos_x, double pos_y, double pos_z)	38
5.6.2.19	isOnSenderLayer(double site_position, enum eCartesianDirection xyz, enum e↔MinMax minmax)	39
5.6.2.20	isOnThisRank(int gi, int gj, int gk, const GridObj &pGrid)	39

5.6.2.21	<code>isOnThisRank(int gl, enum eCartesianDirection xyz, const GridObj &pGrid)</code>	39
5.6.2.22	<code>isOverlapPeriodic(int i, int j, int k, const GridObj &pGrid)</code>	40
5.6.2.23	<code>linspace(double min, double max, int n)</code>	40
5.6.2.24	<code>local_to_global(int i, int j, int k, GridObj *g, std::vector< NumType > &globals)</code>	40
5.6.2.25	<code>matrix_multiply(const std::vector< std::vector< double > > &A, const std::vector< double > &x)</code>	41
5.6.2.26	<code>onespace(int min, int max)</code>	41
5.6.2.27	<code>stridedCopy(NumType *dest, NumType *src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset=0)</code>	41
5.6.2.28	<code>subtract(std::vector< double > a, std::vector< double > b)</code>	42
5.6.2.29	<code>upToZero(NumType x)</code>	42
5.6.2.30	<code>vecmultiply(double scalar, std::vector< double > vec)</code>	42
5.6.2.31	<code>vecnorm(double vec[L_dims])</code>	42
5.6.2.32	<code>vecnorm(double val1, double val2)</code>	43
5.6.2.33	<code>vecnorm(double val1, double val2, double val3)</code>	43
5.6.2.34	<code>vecnorm(std::vector< double > vec)</code>	43
5.6.2.35	<code>vecnorm(NumType a1, NumType a2, NumType a3)</code>	44
5.6.2.36	<code>vecnorm(NumType a1, NumType a2)</code>	44
5.6.3	Member Data Documentation	44
5.6.3.1	<code>dir_reflect</code>	44
5.6.3.2	<code>logfile</code>	45
5.6.3.3	<code>path_str</code>	45
5.7	IBBody Class Reference	45
5.7.1	Detailed Description	46
5.7.2	Constructor & Destructor Documentation	46
5.7.2.1	<code>IBBody(void)</code>	46
5.7.2.2	<code>~IBBody(void)</code>	46
5.7.2.3	<code>IBBody(GridObj *g)</code>	46
5.7.3	Member Function Documentation	47
5.7.3.1	<code>addMarker(double x, double y, double z, bool flex_rigid)</code>	47
5.7.3.2	<code>makeBody(double radius, std::vector< double > centre, bool flex_rigid, bool moving, int group)</code>	47

5.7.3.3	<code>makeBody(std::vector< double > width_length_depth, std::vector< double > angles, std::vector< double > centre, bool flex_rigid, bool deform, int group)</code>	47
5.7.3.4	<code>makeBody(int numbermarkers, std::vector< double > start_point, double fil_↵length, std::vector< double > angles, std::vector< int > BCs, bool flex_rigid, bool deform, int group)</code>	48
5.7.3.5	<code>makeBody(std::vector< double > width_length, double angle, std::vector< double > centre, bool flex_rigid, bool deform, int group, bool plate)</code>	48
5.7.3.6	<code>makeBody(PCpts *_PCpts)</code>	48
5.7.4	Friends And Related Function Documentation	48
5.7.4.1	<code>ObjectManager</code>	48
5.7.5	Member Data Documentation	49
5.7.5.1	<code>BCs</code>	49
5.7.5.2	<code>deformable</code>	49
5.7.5.3	<code>delta_rho</code>	49
5.7.5.4	<code>flex_rigid</code>	49
5.7.5.5	<code>flexural_rigidity</code>	49
5.7.5.6	<code>groupID</code>	49
5.7.5.7	<code>tension</code>	49
5.8	IBMarker Class Reference	49
5.8.1	Detailed Description	50
5.8.2	Constructor & Destructor Documentation	50
5.8.2.1	<code>IBMarker(void)</code>	50
5.8.2.2	<code>~IBMarker(void)</code>	51
5.8.2.3	<code>IBMarker(double xPos, double yPos, double zPos, bool flex_rigid=false)</code>	51
5.8.3	Friends And Related Function Documentation	51
5.8.3.1	<code>IBBody</code>	51
5.8.3.2	<code>ObjectManager</code>	51
5.8.4	Member Data Documentation	51
5.8.4.1	<code>deltaval</code>	51
5.8.4.2	<code>desired_vel</code>	51
5.8.4.3	<code>dilation</code>	51
5.8.4.4	<code>epsilon</code>	51

5.8.4.5	flex_rigid	51
5.8.4.6	fluid_vel	52
5.8.4.7	force_xyz	52
5.8.4.8	local_area	52
5.8.4.9	position_old	52
5.9	IVector< GenTyp > Class Template Reference	52
5.9.1	Detailed Description	53
5.9.2	Constructor & Destructor Documentation	53
5.9.2.1	IVector()	53
5.9.2.2	~IVector()	53
5.9.2.3	IVector(size_t size, GenTyp val)	53
5.9.3	Member Function Documentation	53
5.9.3.1	operator()(size_t i, size_t j, size_t k, size_t v, size_t j_max, size_t k_max, size_t v_max)	53
5.9.3.2	operator()(size_t i, size_t j, size_t k, size_t j_max, size_t k_max)	54
5.9.3.3	operator()(size_t i, size_t j, size_t j_max)	54
5.10	MpiManager::layer_edges Struct Reference	54
5.10.1	Detailed Description	55
5.10.2	Member Data Documentation	55
5.10.2.1	X	55
5.10.2.2	Y	55
5.10.2.3	Z	55
5.11	Marker Class Reference	55
5.11.1	Detailed Description	56
5.11.2	Constructor & Destructor Documentation	56
5.11.2.1	Marker(void)	56
5.11.2.2	~Marker(void)	56
5.11.2.3	Marker(double x, double y, double z)	56
5.11.3	Member Data Documentation	57
5.11.3.1	position	57
5.11.3.2	supp_i	57

5.11.3.3	supp_j	57
5.11.3.4	supp_k	57
5.11.3.5	support_rank	57
5.12	MarkerData Class Reference	57
5.12.1	Detailed Description	58
5.12.2	Constructor & Destructor Documentation	58
5.12.2.1	MarkerData(int i, int j, int k, double x, double y, double z, int ID)	58
5.12.2.2	MarkerData(void)	58
5.12.2.3	~MarkerData(void)	58
5.12.3	Member Data Documentation	59
5.12.3.1	i	59
5.12.3.2	ID	59
5.12.3.3	j	59
5.12.3.4	k	59
5.12.3.5	x	59
5.12.3.6	y	59
5.12.3.7	z	59
5.13	MpiManager Class Reference	59
5.13.1	Detailed Description	62
5.13.2	Member Function Documentation	62
5.13.2.1	destroyInstance()	62
5.13.2.2	getInstance()	62
5.13.2.3	mpi_buffer_pack(int dir, GridObj *g)	62
5.13.2.4	mpi_buffer_size()	62
5.13.2.5	mpi_buffer_size_recv(GridObj *&g)	62
5.13.2.6	mpi_buffer_size_send(GridObj *&g)	63
5.13.2.7	mpi_buffer_unpack(int dir, GridObj *g)	63
5.13.2.8	mpi_buildCommunicators()	63
5.13.2.9	mpi_communicate(int level, int regnum)	63
5.13.2.10	mpi_getOpposite(int direction)	64

5.13.2.11	<code>mpi_gridbuild()</code>	64
5.13.2.12	<code>mpi_init()</code>	64
5.13.2.13	<code>mpi_writeout_buf(std::string filename, int dir)</code>	64
5.13.3	Member Data Documentation	64
5.13.3.1	<code>buffer_recv_info</code>	64
5.13.3.2	<code>buffer_send_info</code>	64
5.13.3.3	<code>f_buffer_recv</code>	64
5.13.3.4	<code>f_buffer_send</code>	65
5.13.3.5	<code>global_dims</code>	65
5.13.3.6	<code>global_edge_ind</code>	65
5.13.3.7	<code>global_edge_pos</code>	65
5.13.3.8	<code>Grids</code>	65
5.13.3.9	<code>local_size</code>	65
5.13.3.10	<code>logout</code>	65
5.13.3.11	<code>MPI_cartlab</code>	65
5.13.3.12	<code>MPI_coords</code>	66
5.13.3.13	<code>MPI_dims</code>	66
5.13.3.14	<code>my_rank</code>	66
5.13.3.15	<code>neighbour_coords</code>	66
5.13.3.16	<code>neighbour_rank</code>	66
5.13.3.17	<code>num_ranks</code>	66
5.13.3.18	<code>p_data</code>	66
5.13.3.19	<code>recv_layer_pos</code>	66
5.13.3.20	<code>recv_stat</code>	66
5.13.3.21	<code>send_requests</code>	66
5.13.3.22	<code>send_stat</code>	67
5.13.3.23	<code>sender_layer_pos</code>	67
5.13.3.24	<code>subGrid_comm</code>	67
5.13.3.25	<code>world_comm</code>	67
5.14	ObjectManager Class Reference	67

5.14.1 Detailed Description	69
5.14.2 Member Function Documentation	69
5.14.2.1 bfl_build_body(int body_type)	69
5.14.2.2 bfl_build_body(PCpts *_PCpts)	69
5.14.2.3 computeLiftDrag(int i, int j, int k, GridObj *g)	70
5.14.2.4 destroyInstance()	70
5.14.2.5 getInstance()	70
5.14.2.6 getInstance(GridObj *g)	70
5.14.2.7 ibm_apply()	70
5.14.2.8 ibm_banbks(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double b[])	70
5.14.2.9 ibm_bandec(double **a, long n, int m1, int m2, double **al, unsigned long indx[], double *d)	71
5.14.2.10 ibm_bicgstab(std::vector< std::vector< double > > &Amatrix, std::vector< double > &bVector, std::vector< double > &epsilon, double tolerance, int maxiterations)	71
5.14.2.11 ibm_build_body(int body_type)	72
5.14.2.12 ibm_build_body(PCpts *_PCpts, GridObj *owner)	72
5.14.2.13 ibm_computeforce(int ib)	72
5.14.2.14 ibm_deltakernel(double rad, double dilation)	72
5.14.2.15 ibm_findepsilon(int ib)	73
5.14.2.16 ibm_findsupport(int ib, int m)	73
5.14.2.17 ibm_initialise()	73
5.14.2.18 ibm_interpol(int ib)	73
5.14.2.19 ibm_jacowire(int ib)	74
5.14.2.20 ibm_move_bodies()	74
5.14.2.21 ibm_position_update(int ib)	74
5.14.2.22 ibm_position_update_grp(int group)	74
5.14.2.23 ibm_spread(int ib)	74
5.14.2.24 io_readInCloud(PCpts *_PCpts, eObjectType objtype)	75
5.14.2.25 io_restart(bool IO_flag, int level)	75
5.14.2.26 io_vtk_IBwriter(double tval)	75
5.14.2.27 io_write_body_pos(int timestep)	75

5.14.2.28 <code>io_write_lift_drag(int timestep)</code>	75
5.14.2.29 <code>io_writeForceOnObject(double tval)</code>	76
5.14.3 Friends And Related Function Documentation	76
5.14.3.1 <code>GridObj</code>	76
5.15 <code>PCpts</code> Class Reference	76
5.15.1 Detailed Description	77
5.15.2 Constructor & Destructor Documentation	77
5.15.2.1 <code>PCpts(void)</code>	77
5.15.2.2 <code>~PCpts(void)</code>	77
5.15.3 Member Data Documentation	77
5.15.3.1 <code>x</code>	77
5.15.3.2 <code>y</code>	77
5.15.3.3 <code>z</code>	77
5.16 <code>MpiManager::phdf5_struct</code> Struct Reference	78
5.16.1 Detailed Description	78
5.16.2 Member Data Documentation	78
5.16.2.1 <code>halo_max</code>	78
5.16.2.2 <code>halo_min</code>	78
5.16.2.3 <code>i_end</code>	79
5.16.2.4 <code>i_start</code>	79
5.16.2.5 <code>j_end</code>	79
5.16.2.6 <code>j_start</code>	79
5.16.2.7 <code>k_end</code>	79
5.16.2.8 <code>k_start</code>	79
5.16.2.9 <code>level</code>	79
5.16.2.10 <code>region</code>	79
5.16.2.11 <code>writable_data_count</code>	79

6	File Documentation	81
6.1	BFLBody.cpp File Reference	81
6.2	BFLBody.h File Reference	81
6.3	BFLMarker.cpp File Reference	81
6.4	BFLMarker.h File Reference	82
6.5	Body.h File Reference	82
6.6	definitions.h File Reference	82
6.6.1	Macro Definition Documentation	87
6.6.1.1	L_a_x	87
6.6.1.2	L_a_y	87
6.6.1.3	L_a_z	87
6.6.1.4	L_a_z	88
6.6.1.5	L_b_x	88
6.6.1.6	L_b_y	88
6.6.1.7	L_b_z	88
6.6.1.8	L_b_z	88
6.6.1.9	L_bfl_length	88
6.6.1.10	L_bfl_length_ref	88
6.6.1.11	L_bfl_on_grid_lev	88
6.6.1.12	L_bfl_on_grid_reg	88
6.6.1.13	L_bfl_scale_direction	88
6.6.1.14	L_block_on_grid_lev	89
6.6.1.15	L_block_on_grid_reg	89
6.6.1.16	L_block_x_max	89
6.6.1.17	L_block_x_min	89
6.6.1.18	L_block_y_max	89
6.6.1.19	L_block_y_min	89
6.6.1.20	L_block_z_max	89
6.6.1.21	L_block_z_max	89
6.6.1.22	L_block_z_min	89

6.6.1.23	L_block_z_min	89
6.6.1.24	L_centre_bfl_z	90
6.6.1.25	L_centre_bfl_z	90
6.6.1.26	L_centre_ibb_z	90
6.6.1.27	L_centre_ibb_z	90
6.6.1.28	L_centre_object_z	90
6.6.1.29	L_centre_object_z	90
6.6.1.30	L_dims	90
6.6.1.31	L_end_BC	90
6.6.1.32	L_grav_direction	90
6.6.1.33	L_grav_force	90
6.6.1.34	L_HDF5_OUTPUT	91
6.6.1.35	L_IB_Lev	91
6.6.1.36	L_IB_Reg	91
6.6.1.37	L_ibb_angle_horz	91
6.6.1.38	L_ibb_angle_vert	91
6.6.1.39	L_ibb_d	91
6.6.1.40	L_ibb_d	91
6.6.1.41	L_ibb_deform	91
6.6.1.42	L_ibb_delta_rho	91
6.6.1.43	L_ibb_EI	91
6.6.1.44	L_ibb_filament_length	92
6.6.1.45	L_ibb_filament_start_x	92
6.6.1.46	L_ibb_filament_start_y	92
6.6.1.47	L_ibb_filament_start_z	92
6.6.1.48	L_ibb_flex_rigid	92
6.6.1.49	L_ibb_I	92
6.6.1.50	L_ibb_length	92
6.6.1.51	L_ibb_length_ref	92
6.6.1.52	L_ibb_on_grid_lev	92

6.6.1.53	L_abb_on_grid_reg	92
6.6.1.54	L_abb_r	93
6.6.1.55	L_abb_scale_direction	93
6.6.1.56	L_abb_w	93
6.6.1.57	L_abb_x	93
6.6.1.58	L_abb_y	93
6.6.1.59	L_abb_z	93
6.6.1.60	L_IBM_ON	93
6.6.1.61	L_INLET_ON	93
6.6.1.62	L_INLET_REGULARISED	93
6.6.1.63	L_INSERT_CIRCLE_SPHERE	93
6.6.1.64	L_K	93
6.6.1.65	L_K	94
6.6.1.66	L_M	94
6.6.1.67	L_MPI_dir	94
6.6.1.68	L_N	94
6.6.1.69	L_NO_FLOW	94
6.6.1.70	L_num_markers	94
6.6.1.71	L_NumLev	94
6.6.1.72	L_NumReg	94
6.6.1.73	L_nVels	94
6.6.1.74	L_object_length	94
6.6.1.75	L_object_length_ref	94
6.6.1.76	L_object_on_grid_lev	95
6.6.1.77	L_object_on_grid_reg	95
6.6.1.78	L_object_scale_direction	95
6.6.1.79	L_out_every	95
6.6.1.80	L_out_every_forces	95
6.6.1.81	L_out_every_probe	95
6.6.1.82	L_OUTLET_ON	95

6.6.1.83	L_output_precision	95
6.6.1.84	L_PARABOLIC_INLET	95
6.6.1.85	L_PI	95
6.6.1.86	L_Re	96
6.6.1.87	L_restart_out_every	96
6.6.1.88	L_rho_in	96
6.6.1.89	L_start_BC	96
6.6.1.90	L_start_bfl_x	96
6.6.1.91	L_start_bfl_y	96
6.6.1.92	L_start_ibt_x	96
6.6.1.93	L_start_ibt_y	96
6.6.1.94	L_start_object_x	96
6.6.1.95	L_start_object_y	96
6.6.1.96	L_Timesteps	97
6.6.1.97	L_u_0x	97
6.6.1.98	L_u_0x	97
6.6.1.99	L_u_0y	97
6.6.1.100	L_u_0y	97
6.6.1.101	L_u_0z	97
6.6.1.102	L_u_0z	97
6.6.1.103	L_u_0z	97
6.6.1.104	L_u_max	97
6.6.1.105	L_u_ref	97
6.6.1.106	L_USE_INLET_PROFILE	98
6.6.1.107	L_VTK_BODY_WRITE	98
6.6.1.108	L_wall_thickness_back	98
6.6.1.109	L_wall_thickness_bottom	98
6.6.1.110	L_wall_thickness_front	98
6.6.1.111	L_wall_thickness_top	98
6.6.1.112	L_WALLS_ON	98

6.6.1.113 L_WALLS_ON_2D	98
6.6.1.114 L_Xcores	98
6.6.1.115 L_Ycores	98
6.6.1.116 L_Zcores	99
6.6.1.117 LUMA_VERSION	99
6.6.2 Variable Documentation	99
6.6.2.1 nProbes	99
6.6.2.2 RefXend	99
6.6.2.3 RefXstart	99
6.6.2.4 RefYend	99
6.6.2.5 RefYstart	99
6.6.2.6 RefZend	99
6.6.2.7 RefZstart	99
6.6.2.8 xProbeLims	99
6.6.2.9 yProbeLims	99
6.6.2.10 zProbeLims	99
6.7 GridObj.cpp File Reference	100
6.8 GridObj.h File Reference	100
6.8.1 Enumeration Type Documentation	100
6.8.1.1 eBCType	100
6.8.1.2 eType	101
6.9 GridObj_init_grids.cpp File Reference	101
6.10 GridObj_ops_boundary.cpp File Reference	101
6.11 GridObj_ops_io.cpp File Reference	101
6.12 GridObj_ops_lbm.cpp File Reference	102
6.13 GridUtils.cpp File Reference	102
6.14 GridUtils.h File Reference	102
6.14.1 Enumeration Type Documentation	102
6.14.1.1 eCartesianDirection	102
6.14.1.2 eMinMax	103

6.15	hdf5luma.h File Reference	103
6.15.1	Macro Definition Documentation	104
6.15.1.1	H5_BUILT_AS_DYNAMIC_LIB	104
6.15.1.2	HDF5_EXT_SZIP	104
6.15.1.3	HDF5_EXT_ZLIB	104
6.15.2	Enumeration Type Documentation	104
6.15.2.1	eHdf5SlabType	104
6.15.3	Function Documentation	104
6.15.3.1	hdf5_writeDataSet(hid_t &memspace, hid_t &filespace, hid_t &dataset_id, e↵ Hdf5SlabType slab_type, int N_lim, int M_lim, int K_lim, int N_mod, int M_mod, int K_mod, GridObj *g, T *data, hid_t hdf_datatype, int TL_thickness, MPIManager↵ ::phdf5_struct hdf_data)	104
6.16	IBBody.cpp File Reference	105
6.17	IBBody.h File Reference	105
6.18	IBMarker.cpp File Reference	105
6.19	IBMarker.h File Reference	105
6.20	IVector.h File Reference	105
6.21	main_lbm.cpp File Reference	106
6.21.1	Function Documentation	106
6.21.1.1	main(int argc, char *argv[])	106
6.22	Marker.h File Reference	106
6.23	Mpi_buffer_pack.cpp File Reference	106
6.24	Mpi_buffer_size_recv.cpp File Reference	107
6.25	Mpi_buffer_size_send.cpp File Reference	107
6.26	Mpi_buffer_unpk.cpp File Reference	107
6.27	MpiManager.cpp File Reference	107
6.28	MpiManager.h File Reference	107
6.28.1	Macro Definition Documentation	108
6.28.1.1	range_i_left	108
6.28.1.2	range_i_right	108
6.28.1.3	range_j_down	108
6.28.1.4	range_j_up	108

6.28.1.5	range_k_back	109
6.28.1.6	range_k_front	109
6.29	ObjectManager.cpp File Reference	109
6.30	ObjectManager.h File Reference	109
6.30.1	Enumeration Type Documentation	109
6.30.1.1	eObjectType	109
6.31	ObjectManager_init_bflbody.cpp File Reference	110
6.32	ObjectManager_init_ibmbody.cpp File Reference	110
6.33	ObjectManager_ops_ibm.cpp File Reference	110
6.34	ObjectManager_ops_ibmflex.cpp File Reference	110
6.34.1	Macro Definition Documentation	110
6.34.1.1	SWAP	110
6.34.1.2	SWAP	111
6.34.1.3	TINY	111
6.35	ObjectManager_ops_io.cpp File Reference	111
6.36	PCpts.h File Reference	111
6.37	stdafx.cpp File Reference	111
6.37.1	Variable Documentation	112
6.37.1.1	c	112
6.37.1.2	cs	112
6.37.1.3	w	112
6.38	stdafx.h File Reference	112
6.38.1	Macro Definition Documentation	113
6.38.1.1	L_IS_NAN	113
6.38.1.2	LUMA_FAILED	113
6.38.2	Variable Documentation	113
6.38.2.1	c	113
6.38.2.2	cs	113
6.38.2.3	w	113

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	45
MpiManager::buffer_struct	16
GridObj	17
GridUtils	30
MpiManager::layer_edges	54
Marker	55
BFLMarker	11
IBMarker	49
MarkerData	57
MpiManager	59
ObjectManager	67
PCpts	76
MpiManager::phdf5_struct	78
vector	
IVector< GenTyp >	52
IVector< double >	52
IVector< eType >	52

Chapter 3

Class Index

3.1 Class List

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

BFLBody	
BFL body	9
BFLMarker	
BFL marker	11
Body< MarkerType >	
Generic body class	13
MpiManager::buffer_struct	
Structure storing buffers sizes in each direction for particular grid	16
GridObj	
Grid class	17
GridUtils	
Grid utility class	30
IBBody	
Immersed boundary body	45
IBMarker	
Immersed boundary marker	49
IVector< GenTyp >	
Index-collapsing vector class	52
MpiManager::layer_edges	
Structure containing global positions of the edges of halos	54
Marker	
Generic marker class	55
MarkerData	
Container class to hold marker information	57
MpiManager	
MPI Manager class	59
ObjectManager	
Object Manager class	67
PCpts	
Class to hold point cloud data	76
MpiManager::phdf5_struct	
Structure for storing halo information for HDF5	78

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

BFLBody.cpp	81
BFLBody.h	81
BFLMarker.cpp	81
BFLMarker.h	82
Body.h	82
definitions.h	82
GridObj.cpp	100
GridObj.h	100
GridObj_init_grids.cpp	101
GridObj_ops_boundary.cpp	101
GridObj_ops_io.cpp	101
GridObj_ops_lbm.cpp	102
GridUtils.cpp	102
GridUtils.h	102
hdf5luma.h	103
IBBody.cpp	105
IBBody.h	105
IBMarker.cpp	105
IBMarker.h	105
IVector.h	105
main_lbm.cpp	106
Marker.h	106
Mpi_buffer_pack.cpp	106
Mpi_buffer_size_recv.cpp	107
Mpi_buffer_size_send.cpp	107
Mpi_buffer_unpk.cpp	107
MpiManager.cpp	107
MpiManager.h	107
ObjectManager.cpp	109
ObjectManager.h	109
ObjectManager_init_bflbody.cpp	110
ObjectManager_init_ibmbody.cpp	110
ObjectManager_ops_ibm.cpp	110
ObjectManager_ops_ibmflex.cpp	110
ObjectManager_ops_io.cpp	111
PCpts.h	111
stdafx.cpp	111
stdafx.h	112

Chapter 5

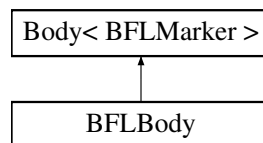
Class Documentation

5.1 BFLBody Class Reference

BFL body.

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

Inheritance diagram for BFLBody:



Public Member Functions

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

Protected Member Functions

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

Protected Attributes

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

Friends

- class [GridObj](#)

5.1.1 Detailed Description

BFL body.

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

5.1.2 Constructor & Destructor Documentation

5.1.2.1 BFLBody::BFLBody (void)

Default constructor.

5.1.2.2 BFLBody::~~BFLBody (void)

Default destructor.

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

Custom constructor to populate body from array of points.

Parameters

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

5.1.3 Member Function Documentation

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

Routine to compute wall distance Q.

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

Parameters

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

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

Routine to compute wall distance Q.

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

Parameters

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

5.1.4 Friends And Related Function Documentation

5.1.4.1 `friend class GridObj` `[friend]`

5.1.5 Member Data Documentation

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

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

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

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

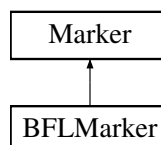
- [BFLBody.h](#)
- [BFLBody.cpp](#)

5.2 BFLMarker Class Reference

BFL marker.

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

Inheritance diagram for BFLMarker:



Public Member Functions

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

Friends

- class [BFLBody](#)

Additional Inherited Members

5.2.1 Detailed Description

BFL marker.

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

5.2.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

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

Custom constructor with position.

Parameters

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

5.2.3 Friends And Related Function Documentation

5.2.3.1 friend class BFLBody [friend]

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

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

5.3 Body< MarkerType > Class Template Reference

Generic body class.

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

Public Member Functions

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

Protected Member Functions

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

Protected Attributes

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

5.3.1 Detailed Description

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

Generic body class.

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

5.3.2 Constructor & Destructor Documentation

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

Default Constructor.

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

Default destructor.

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

Custom constructor setting owning grid.

Parameters

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

5.3.3 Member Function Documentation

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

Add marker to the body.

Parameters

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

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

Retrieve marker data.

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

Parameters

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

Returns

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

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

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

Parameters

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

Returns

true of false

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

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

Parameters

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

Returns

true of false

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

Downsampling marker adding method.

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

Parameters

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

5.3.4 Member Data Documentation

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

Pointer to owning grid.

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

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

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

Array of markers which make up the body.

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

Spacing of the markers in physical units.

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

- [Body.h](#)

5.4 MpiManager::buffer_struct Struct Reference

Structure storing buffers sizes in each direction for particular grid.

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

Public Attributes

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

5.4.1 Detailed Description

Structure storing buffers sizes in each direction for particular grid.

5.4.2 Member Data Documentation

5.4.2.1 `int` `MpiManager::buffer_struct::level`

Grid level.

5.4.2.2 `int` `MpiManager::buffer_struct::region`

Region number.

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

Buffer sizes for each direction.

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

- [MpiManager.h](#)

5.5 GridObj Class Reference

Grid class.

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

Public Member Functions

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

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

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

Public Attributes

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

Friends

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

5.5.1 Detailed Description

Grid class.

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

5.5.2 Constructor & Destructor Documentation

5.5.2.1 GridObj::GridObj (void)

Default Constructor.

5.5.2.2 GridObj::GridObj (int *level*)

Constructor for top level grid.

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

Parameters

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

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

Constructor for a sub-grid.

Parameters

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

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

MPI constructor for top level grid.

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

Parameters

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

5.5.2.5 GridObj::~GridObj (void)

Default Destructor.

5.5.3 Member Function Documentation

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

Method to apply BFL bounce-back.

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

Parameters

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

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

Method to apply half-way bounce-back.

Parameters

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

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

Method to apply extrapolation outlet.

Can only be applied on right-hand wall.

Parameters

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

Parameters

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

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

Method to apply NRBC.

Not implemented in this version.

Parameters

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

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

Method to apply regularised velocity inlet.

Can be applied on any wall.

Parameters

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

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

Method to apply half-way specular reflection.

Symmetry boundary condition for free-slip walls.

Parameters

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

5.5.3.7 void GridObj::bc_solidSiteReset ()

Helper method to set macroscopic quantities of solid sites.

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

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

HDF5 writer.

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

Parameters

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

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

ASCII dump of grid data.

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

Parameters

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

5.5.3.10 void GridObj::io_probeOutput ()

Probe writer.

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

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

Restart file read-writer.

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

Parameters

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

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

Verbose ASCII writer.

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

Parameters

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

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

Wrapper method to add sub-grid to this grid.

Parameters

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

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

Method to apply boundary conditions on lattice.

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

Parameters

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

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

Coalesce operation for pulling information from finer grids.

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

Parameters

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

5.5.3.16 void GridObj::LBM_collide ()

Apply collision operator.

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

Equilibrium calculation.

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

Parameters

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

Returns

equilibrium function.

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

Explosion operation for pushing information to finer grids.

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

Parameters

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

5.5.3.19 void GridObj::LBM_forceGrid ()

Method to compute body forces.

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

5.5.3.20 void GridObj::LBM_init_getInletProfile ()

Method to import an input profile from a file.

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

5.5.3.21 void GridObj::LBM_initBoundLab ()

Method to initialise wall and object labels on L0.

The virtual wind tunnel definitions are implemented by this method.

5.5.3.22 void GridObj::LBM_initGrid ()

Wrapper to initialise all L0 lattice quantities.

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

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

Method to initialise all L0 lattice quantities.

Parameters

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

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

Method to initialise all labels on sub-grids.

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

Parameters

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

5.5.3.25 void GridObj::LBM_initRho ()

Method to initialise the lattice density.

5.5.3.26 void GridObj::LBM_initSolidLab ()

Method to initialise label-based solids.

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

Method to initialise all sub-grid quantities.

Parameters

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

5.5.3.28 void GridObj::LBM_initVelocity ()

Method to initialise the lattice velocity.

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

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

KBC collision operator.

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

Parameters

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

5.5.3.30 void GridObj::LBM_macro ()

Macroscopic update.

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

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

Site-specific macroscopic update.

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

Parameters

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

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

LBM multi-grid kernel.

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

Parameters

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

5.5.3.33 void GridObj::LBM_resetForces ()

Method to reset body forces.

Resets both Cartesian and Lattice force vectors to zero.

5.5.3.34 void GridObj::LBM_stream ()

Streaming operator.

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

5.5.4 Friends And Related Function Documentation

5.5.4.1 friend class GridUtils [friend]

5.5.4.2 friend class MpiManager [friend]

5.5.4.3 friend class ObjectManager [friend]

5.5.5 Member Data Documentation

5.5.5.1 double GridObj::dt

Physical time step size.

5.5.5.2 int GridObj::K_lim

Local size of grid in Z-direction.

5.5.5.3 IVector<eType> GridObj::LatTyp

Flattened 3D array of site labels.

5.5.5.4 int GridObj::level

Level in embedded grid hierarchy.

5.5.5.5 int GridObj::M_lim

Local size of grid in Y-direction.

5.5.5.6 int GridObj::N_lim

Local size of grid in X-direction.

5.5.5.7 double GridObj::nu

Kinematic viscosity (in lattice units)

5.5.5.8 double GridObj::omega

Relaxation frequency.

5.5.5.9 int GridObj::t

Number of completed iterations on this level.

5.5.5.10 double GridObj::timeav_mpi_overhead

Time-averaged time of MPI communication.

5.5.5.11 double GridObj::timeav_timestep

Time-averaged time of a timestep.

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

Vector of global X indices of each site.

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

Vector of global X positions of each site.

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

Vector of global Y indices of each site.

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

Vector of global Y positions of each site.

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

Vector of global Z indices of each site.

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

Vector of global Z positions of each site.

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

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

5.6 GridUtils Class Reference

Grid utility class.

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

Static Public Member Functions

- static int [createOutputDirectory](#) (std::string [path_str](#))
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 *&Grids, int level, int region, GridObj *&ptr)
- Get a pointer to a given grid in the hierarchy.*
- static std::vector< int > **getVoxInd** (double x, double y, double z)
- Get global voxel indices.*
- static int **getVoxInd** (double p)
- Get global voxel index.*
- static bool **isOverlapPeriodic** (int i, int j, int k, const GridObj &pGrid)
- Finds out whether halo containng i,j,k links to neighbour rank periodically.*
- static bool **isOnThisRank** (int gi, int gj, int gk, const GridObj &pGrid)
- Finds out whether site with supplied index in on the current rank.*
- static bool **isOnThisRank** (int gl, enum eCartesianDirection xyz, const GridObj &pGrid)
- Finds out whether global index can be found on the current rank.*
- static bool **hasThisSubGrid** (const GridObj &pGrid, int RegNum)
- Finds out whether specified refined region is on 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, enum eCartesianDirection xyz, enum eMinMax minmax)
- Check whether site is on an inner (sender) halo.*
- static bool **isOnRecvLayer** (double site_position, enum eCartesianDirection xyz, enum eMinMax minmax)
- Check whether site is on an outer (receiver) halo.*
- static bool **isOffGrid** (int i, int j, int k, GridObj &g)
- Tests whether a site is on a given grid.*
- template<typename NumType >
static NumType **vecnorm** (NumType a1, NumType a2, NumType a3)
- Computes the L2-norm.*
- template<typename NumType >
static NumType **vecnorm** (NumType a1, NumType a2)
- Computes the L2-norm.*
- template<typename NumType >
static NumType **upToZero** (NumType x)
- Rounds a negative value up to zero.*
- template<typename NumType >
static NumType **downToLimit** (NumType x, NumType limit)

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

- `template<typename NumType >`
`static NumType factorial (NumType n)`
Computes the factorial of the supplied value.
- `template<typename NumType >`
`static void stridedCopy (NumType *dest, NumType *src, size_t block, size_t offset, size_t stride, size_t count, size_t buf_offset=0)`
Performs a strided memcopy.
- `template<typename NumType >`
`static void global_to_local (int i, int j, int k, GridObj *g, std::vector< NumType > &locals)`
Maps global indices to local indices.
- `template<typename NumType >`
`static void local_to_global (int i, int j, int k, GridObj *g, std::vector< NumType > &globals)`
Maps local indices to global indices.

Static Public Attributes

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

5.6.1 Detailed Description

Grid utility class.

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

5.6.2 Member Function Documentation

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

Adds two vectors.

Parameters

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

Returns

vector which is $a + b$.

5.6.2.2 `int 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 9 if the directory creation was not attempted due to not being rank 0. Returns platform specific codes for everything else.

Parameters

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

Returns

indicator of status of action.

5.6.2.3 `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.

5.6.2.4 `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.

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

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

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

Parameters

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

Returns

NumType rounded value

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

Computes the factorial of the supplied value.

If *n* == 0 then returns 1.

Parameters

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

Returns

NumType *n* factorial

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

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.

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

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.

5.6.2.9 `void GridUtils::getGrid (GridObj *& 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>	x-position of site.
	<i>level</i>	y-position of site.
	<i>region</i>	z-position of site.
out	<i>ptr</i>	pointer containing address of grid in hierarchy.

5.6.2.10 `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.

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

Get global voxel indices.

Will return the voxel indices of the nearest voxel on the lattice for a given point in global space. Assumes that the physical lattice size is 1 unit big which relies on the position being scaled. If not, method acts as a natural voxel grid filter.

Parameters

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

Returns

vector of indices of the nearest voxel.

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

Get global voxel index.

Will return the voxel indices of the nearest voxel on the lattice for a given point in global space. Assumes that the physical lattice size is 1 unit big which relies on the position being scaled. If not, method acts as a natural voxel grid filter.

Parameters

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

Returns

corresponding global index.

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

Maps global indices to local indices.

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

Parameters

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

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

Finds out whether specified refined region is on the grid provided.

Parameters

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

Returns

boolean answer.

5.6.2.15 `bool GridUtils::isOffGrid (int i, int j, int k, GridObj & 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.

5.6.2.16 `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_x</i>	x-position of site.
--------------	---------------------

Parameters

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

Returns

boolean answer.

5.6.2.17 `bool GridUtils::isOnRecvLayer (double site_position, enum eCartesianDirection dir, enum eMinMax maxmin)`
`[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>dir</i>	cartesian direction.
<i>maxmin</i>	choice of edge in given direction.

Returns

boolean answer.

5.6.2.18 `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</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.

5.6.2.19 `bool GridUtils::isOnSenderLayer (double site_position, enum eCartesianDirection dir, enum eMinMax maxmin) [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>dir</i>	cartesian direction.
<i>maxmin</i>	choice of edge in given direction.

Returns

boolean answer.

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

Finds out whether site with supplied index in on the current rank.

Parameters

<i>gi</i>	global i-index of site.
<i>gj</i>	global j-index of site.
<i>gk</i>	global k-index of site.
<i>grid</i>	grid being queried.

Returns

boolean answer.

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

Finds out whether global index can be found on the current rank.

Parameters

<i>gl</i>	global index (i,j or k).
<i>xyz</i>	cartesian direction of interest.
<i>grid</i>	grid being queried.

Returns

boolean answer.

5.6.2.22 `bool GridUtils::isOverlapPeriodic (int i, int j, int k, const GridObj & 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.

5.6.2.23 `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*.

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

Maps local indices to global indices.

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

Parameters

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

5.6.2.25 `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 $A * x$.

5.6.2.26 `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.

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

Performs a strided memcpy.

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

Parameters

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

5.6.2.28 `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.

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

Rounds a negative value up to zero.

If value is positive, return the value unchanged.

Parameters

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

Returns

NumType rounded value

5.6.2.30 `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.

5.6.2.31 `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.

5.6.2.32 `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.

5.6.2.33 `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.

5.6.2.34 `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.

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

Computes the L2-norm.

Parameters

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

Returns

NumType scalar quantity

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

Computes the L2-norm.

Parameters

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

Returns

NumType scalar quantity

5.6.3 Member Data Documentation

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

Initial value:

```
=
{
    {1, 0, 2, 3, 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.

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

Handle to output file.

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

Static string representing output path.

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

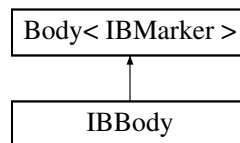
- [GridUtils.h](#)
- [GridObj.cpp](#)
- [GridUtils.cpp](#)
- [main_ibm.cpp](#)

5.7 IBody Class Reference

Immersed boundary body.

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

Inheritance diagram for IBody:



Public Member Functions

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

Protected Attributes

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

Friends

- class [ObjectManager](#)

Additional Inherited Members

5.7.1 Detailed Description

Immersed boundary body.

5.7.2 Constructor & Destructor Documentation

5.7.2.1 `IBBody::IBBody (void)`

Constructor which sets group ID to zero by default.

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

Default destructor.

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

Constructor which assigns the owner grid.

Also sets the group ID to zero.

Parameters

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

5.7.3 Member Function Documentation

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

Method to add an IB marker to the body.

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

Parameters

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

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

Method to seed markers for a sphere / circle.

Parameters

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

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

Method to seed markers for a cuboid / rectangle.

Parameters

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

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

Method to seed markers for a flexible filament.

Parameters

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

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

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

Parameters

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

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

Method to build a body from a point cloud.

Flexibility and deformable properties taken from definitions.

Parameters

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

5.7.4 Friends And Related Function Documentation

5.7.4.1 friend class ObjectManager [friend]

5.7.5 Member Data Documentation

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

BCs type flags (flexible bodies)

5.7.5.2 `bool IBBody::deformable` [protected]

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

5.7.5.3 `double IBBody::delta_rho` [protected]

Difference in density between fluid and solid in lattice units.

5.7.5.4 `bool IBBody::flex_rigid` [protected]

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

5.7.5.5 `double IBBody::flexural_rigidity` [protected]

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

5.7.5.6 `int IBBody::groupID` [protected]

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

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

Tension between the current marker and its neighbour.

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

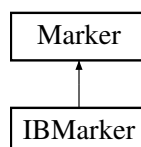
- [IBBody.h](#)
- [IBBody.cpp](#)

5.8 IBMarker Class Reference

Immersed boundary marker.

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

Inheritance diagram for IBMarker:



Public Member Functions

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

Protected Attributes

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

Friends

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

Additional Inherited Members

5.8.1 Detailed Description

Immersed boundary marker.

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

5.8.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

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

Custom constructor with position.

Parameters

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

5.8.3 Friends And Related Function Documentation

5.8.3.1 friend class IBBody [friend]

5.8.3.2 friend class ObjectManager [friend]

5.8.4 Member Data Documentation

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

Value of delta function for a given support node.

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

Desired velocity at marker.

5.8.4.3 double IBMarker::dilation [protected]

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

5.8.4.4 double IBMarker::epsilon [protected]

Scaling parameter.

5.8.4.5 bool IBMarker::flex_rigid [protected]

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

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

Fluid velocity interpolated from lattice nodes.

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

Restorative force vector on marker.

5.8.4.8 `double IBMarker::local_area` [protected]

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

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

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

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

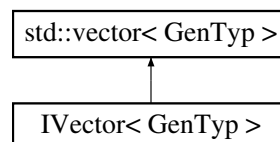
- [IBMarker.h](#)
- [IBMarker.cpp](#)

5.9 `IVector< GenTyp >` Class Template Reference

Index-collapsing vector class.

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

Inheritance diagram for `IVector< GenTyp >`:



Public Member Functions

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

5.9.1 Detailed Description

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

Index-collapsing vector class.

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

5.9.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

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

Custom constructor taking type and value.

Parameters

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

5.9.3 Member Function Documentation

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

4D array index flatten.

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

Parameters

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

Returns

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

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

3D array index flatten.

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

Parameters

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

Returns

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

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

2D array index flatten.

Parameters

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

Returns

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

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

- [IVector.h](#)

5.10 MpiManager::layer_edges Struct Reference

Structure containing global positions of the edges of halos.

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

Public Attributes

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

5.10.1 Detailed Description

Structure containing global positions of the edges of halos.

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

5.10.2 Member Data Documentation

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

X limits.

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

Y limits.

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

Z limits.

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

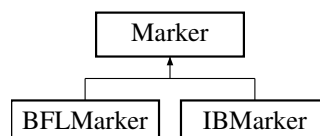
- [MpiManager.h](#)

5.11 Marker Class Reference

Generic marker class.

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

Inheritance diagram for Marker:



Public Member Functions

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

Public Attributes

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

5.11.1 Detailed Description

Generic marker class.

5.11.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

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

Custom constructor which locates marker.

Parameters

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

5.11.3 Member Data Documentation

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

Position vector of marker location in physical units.

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

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

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

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

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

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

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

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

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

- [Marker.h](#)

5.12 MarkerData Class Reference

Container class to hold marker information.

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

Public Member Functions

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

Public Attributes

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

5.12.1 Detailed Description

Container class to hold marker information.

5.12.2 Constructor & Destructor Documentation

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

Constructor.

Parameters

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

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

Default Constructor.

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

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

Default destructor.

5.12.3 Member Data Documentation

5.12.3.1 int MarkerData::i

i-index of primary support site

5.12.3.2 int MarkerData::ID

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

5.12.3.3 int MarkerData::j

j-index of primary support site

5.12.3.4 int MarkerData::k

k-index of primary support site

5.12.3.5 double MarkerData::x

x-position of marker

5.12.3.6 double MarkerData::y

y-position of marker

5.12.3.7 double MarkerData::z

z-position of marker

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

- [Body.h](#)

5.13 MpiManager Class Reference

MPI Manager class.

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

Classes

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

Public Member Functions

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

Static Public Member Functions

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

Public Attributes

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

Static Public Attributes

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

Coordinates in MPI Cartesian topology.

- static `GridObj * Grids`

Pointer to grid hierarchy.

- static `std::ofstream * logout`

Logfile handle.

5.13.1 Detailed Description

MPI Manager class.

Class to manage all MPI aspects of the code.

5.13.2 Member Function Documentation

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

Instance destroyer.

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

Instance creator.

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

Method to pack the communication buffer.

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

Parameters

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

5.13.2.4 `void MpiManager::mpi_buffer_size ()`

Pre-calculation of the buffer sizes.

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

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

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

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

Parameters

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

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

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

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

Parameters

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

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

Method to unpack the communication buffer.

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

Parameters

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

5.13.2.8 int MpiManager::mpi_buildCommunicators ()

Define writable sub-grid communicators.

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

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

Communication routine.

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

Parameters

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

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

Helper method to find opposite direction in MPI topology.

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

Parameters

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

5.13.2.11 void MpiManager::mpi_gridbuild ()

Domain decomposition.

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

5.13.2.12 void MpiManager::mpi_init ()

Initialisation routine.

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

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

Buffer ASCII writer.

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

5.13.3 Member Data Documentation

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

Vectors of buffer_info structures holding receiver layer size info.

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

Vectors of buffer_info structures holding sender layer size info.

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

Array of resizable incoming buffers used for data transfer.

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

Array of resizable outgoing buffers used for data transfer.

5.13.3.5 `int MpiManager::global_dims[3]`

Global dimensions of problem coarse lattice.

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

Global indices of coarse lattice nodes represented on this rank.

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

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

Global positions of coarse lattice nodes represented on this rank.

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

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

Pointer to grid hierarchy.

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

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

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

Logfile handle.

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

Initial value:

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

Cartesian unit vectors pointing to each neighbour in Cartesian topology.

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

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

Coordinates in MPI Cartesian topology.

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

Size of MPI Cartesian topology.

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

Rank number.

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

Coordinates in MPI topology of neighbour ranks.

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

Neighbour rank number for each direction in Cartesian topology.

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

Total number of ranks in MPI Cartesian topology.

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

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

5.13.3.19 `layer_edges` `MpiManager::recv_layer_pos`

Structure containing receiver layer edge positions.

5.13.3.20 `MPI_Status` `MpiManager::recv_stat`

Status structure for Receive return information.

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

Array of request structures for handles to posted ISends.

5.13.3.22 MPI_Status MpiManager::send_stat[L_MPI_dir]

Array of statuses for each lsend.

5.13.3.23 layer_edges MpiManager::sender_layer_pos

Structure containing sender layer edge positions.

5.13.3.24 MPI_Comm MpiManager::subGrid_comm[L_NumLev * L_NumReg]

Communicators for sub-grid / region combinations.

5.13.3.25 MPI_Comm MpiManager::world_comm

Global MPI communicator.

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

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

5.14 ObjectManager Class Reference

Object Manager class.

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

Public Member Functions

- void [ibm_apply](#) ()
Perform IBM procedure.
- void [ibm_build_body](#) (int body_type)
Builds a prefab immersed boundary body.
- void [ibm_build_body](#) (PCpts *_PCpts, [GridObj](#) *owner)
Wrapper for building a body from a point cloud.
- void [ibm_initialise](#) ()
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_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_move_bodies](#) ()
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_position_update](#) (int ib)
Update the position of a deformable iBody.
- void [ibm_position_update_grp](#) (int group)
Update the position of a group of deformable 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_build_body](#) (int body_type)
Prefab body building routine.
- void [bfl_build_body](#) (PCpts *_PCpts)
Wrapper for building BFL body from point cloud.
- void [computeLiftDrag](#) (int i, int j, int k, [GridObj](#) *g)
Compute forces on a rigid object.
- void [io_vtk_IBwriter](#) (double tval)
Write IB body data to VTK file.
- void [io_write_body_pos](#) (int timestep)
Write out position of immersed boundary bodies.
- void [io_write_lift_drag](#) (int timestep)
Write out forces on the markers of immersed boundary bodies.
- void [io_restart](#) (bool IO_flag, int level)
Read/write IB body information to restart file.

- void `io_readInCloud` (`PCpts * _PCpts`, `eObjectType` objtype)
Read in point cloud data.
- void `io_writeForceOnObject` (double tval)
Write out the forces on a solid object.

Static Public Member Functions

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

Friends

- class `GridObj`

5.14.1 Detailed Description

Object Manager class.

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

5.14.2 Member Function Documentation

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

Prefab body building routine.

Not implemented in this version.

Parameters

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

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

Wrapper for building BFL body from point cloud.

Parameters

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

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

Compute forces on a rigid object.

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

Parameters

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

5.14.2.4 void ObjectManager::destroyInstance () [static]

Destroy instance method.

Instance destructor.

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

Get instance method.

Instance creator.

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

Overloaded get instance passing in pointer to grid hierarchy.

Instance creator with grid hierarchy assignment.

Parameters

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

5.14.2.7 void ObjectManager::ibm_apply ()

Perform IBM procedure.

5.14.2.8 void ObjectManager::ibm_banbks (double ** *a*, long *n*, int *m1*, int *m2*, double ** *aI*, 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

5.14.2.9 `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

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

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.

5.14.2.11 void ObjectManager::ibm_build_body (int *body_type*)

Builds a prefab immersed boundary body.

Parameters

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

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

Wrapper for building a body from a point cloud.

Parameters

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

5.14.2.13 void ObjectManager::ibm_computeforce (int *ib*)

Compute restorative force at each marker in a body.

Parameters

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

5.14.2.14 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.

5.14.2.15 double ObjectManager::ibm_findepsilon (int *ib*)

Compute epsilon for a given iBody.

Parameters

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

5.14.2.16 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.

5.14.2.17 void ObjectManager::ibm_initialise ()

Initialise the array of iBodies.

Computes support and epsilon values.

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

Interpolate velocity field onto markers.

Parameters

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

5.14.2.19 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.
-----------	--

5.14.2.20 void ObjectManager::ibm_move_bodies ()

Moves iBodies after applying IBM.

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

5.14.2.21 void ObjectManager::ibm_position_update (int *ib*)

Update the position of a deformable iBody.

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

Parameters

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

5.14.2.22 void ObjectManager::ibm_position_update_grp (int *group*)

Update the position of a group of deformable iBodies.

Updates the position of a group of non-flexible moving (deformable) 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.
--------------	-------------------------

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

Spread restorative force back onto marker support.

Parameters

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

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

Read in point cloud data.

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

Parameters

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

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

Read/write IB body information to restart file.

Parameters

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

5.14.2.26 void ObjectManager::io_vtk_IBwriter (double tval)

Write IB body data to VTK file.

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

Parameters

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

5.14.2.27 void ObjectManager::io_write_body_pos (int timestep)

Write out position of immersed boundary bodies.

Parameters

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

5.14.2.28 void ObjectManager::io_write_lift_drag (int timestep)

Write out forces on the markers of immersed boundary bodies.

Parameters

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

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

Write out the forces on a solid object.

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

Parameters

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

5.14.3 Friends And Related Function Documentation

5.14.3.1 friend class GridObj [friend]

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

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

5.15 PCpts Class Reference

Class to hold point cloud data.

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

Public Member Functions

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

Public Attributes

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

5.15.1 Detailed Description

Class to hold point cloud data.

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

5.15.2 Constructor & Destructor Documentation

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

Default constructor.

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

Default destructor.

5.15.3 Member Data Documentation

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

Vector of X positions.

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

Vector of Y positions.

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

Vector of Z positions.

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

- [PCpts.h](#)

5.16 MpiManager::phdf5_struct Struct Reference

Structure for storing halo information for HDF5.

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

Public Attributes

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

5.16.1 Detailed Description

Structure for storing halo information for HDF5.

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

5.16.2 Member Data Documentation

5.16.2.1 int MpiManager::phdf5_struct::halo_max

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

5.16.2.2 int MpiManager::phdf5_struct::halo_min

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

5.16.2.3 `int MpiManager::phdf5_struct::i_end`

Ending i-index for writable region.

5.16.2.4 `int MpiManager::phdf5_struct::i_start`

Starting i-index for writable region.

5.16.2.5 `int MpiManager::phdf5_struct::j_end`

Ending j-index for writable region.

5.16.2.6 `int MpiManager::phdf5_struct::j_start`

Starting j-index for writable region.

5.16.2.7 `int MpiManager::phdf5_struct::k_end`

Ending k-index for writable region.

5.16.2.8 `int MpiManager::phdf5_struct::k_start`

Starting k-index for writable region.

5.16.2.9 `int MpiManager::phdf5_struct::level`

Grid level to which these data correspond.

5.16.2.10 `int MpiManager::phdf5_struct::region`

Region number to which these data correspond.

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

Writable data count.

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

- [MpiManager.h](#)

Chapter 6

File Documentation

6.1 BFLBody.cpp File Reference

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

6.2 BFLBody.h File Reference

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

Classes

- class [BFLBody](#)
BFL body.

6.3 BFLMarker.cpp File Reference

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

6.4 BFLMarker.h File Reference

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

Classes

- class [BFLMarker](#)

BFL marker.

6.5 Body.h File Reference

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

Classes

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

6.6 definitions.h File Reference

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

Macros

- #define [LUMA_VERSION](#) "1.2.0-alpha"
LUMA version.
- #define [L_PI](#) 3.14159265358979323846
PI definition.
- #define [L_out_every](#) 50
How many timesteps before whole grid output.
- #define [L_out_every_forces](#) 50
Specific output frequency of body forces.

- `#define L_output_precision 6`
Precision of output (for text writers)
- `#define L_HDF5_OUTPUT`
HDF5 dump on output.
- `#define L_out_every_probe 250`
Write out frequency of probe output.
- `#define L_grav_force 0`
Expression for the gravity force.
- `#define L_grav_direction eXDirection`
Gravity direction (specify using enumeration)
- `#define L_NO_FLOW`
Initialise the domain with no flow.
- `#define L_restart_out_every 10000`
Frequency of write out of restart file.
- `#define L_Timesteps 19800`
Number of time steps to run simulation for.
- `#define L_Xcores 2`
Number of MPI ranks to divide domain into in X direction.
- `#define L_Ycores 3`
- `#define L_Zcores 2`
- `#define L_dims 2`
Number of dimensions to the problem.
- `#define L_N (1*220+1)`
Number of x lattice sites.
- `#define L_M (1*41+2)`
Number of y lattice sites.
- `#define L_K 60`
Number of z lattice sites.
- `#define L_a_x -0.005`
Start of domain-x.
- `#define L_b_x 2.205`
End of domain-x.
- `#define L_a_y -0.01`
Start of domain-y.
- `#define L_b_y 0.42`
End of domain-y.
- `#define L_a_z 0`
Start of domain-z.
- `#define L_b_z 6`
End of domain-z.
- `#define L_USE_INLET_PROFILE`
Use an inlet profile.
- `#define L_PARABOLIC_INLET`
Use analytic expression for inlet profile - if not then ASCII file is read (requires L_USE_INLET_PROFILE)
- `#define L_u_ref (0.0707070707070707)`
Reference velocity for scaling, can be mean inlet velocity.
- `#define L_u_max L_u_ref*1.5`
Max velocity of inlet profile.
- `#define L_u_0x L_u_ref`
Initial/inlet x-velocity.
- `#define L_u_0y 0`

- Initial/inlet y-velocity.*
- #define `L_u_0z` 0
- Initial/inlet z-velocity.*
- #define `L_rho_in` 1
- Initial density.*
- #define `L_Re` 20
- Desired Reynolds number.*
- #define `L_IBM_ON`
- Turn on IBM.*
- #define `L_IB_Lev` 2
- Grid level for immersed boundary object (0 if no refined regions, -1 if no IBM)*
- #define `L_IB_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_on_grid_lev` `L_IB_Lev`
- Provide grid level on which object should be added.*
- #define `L_ibb_on_grid_reg` `L_IB_Reg`
- Provide grid region on which object should be added.*
- #define `L_start_ibb_x` 0.3
- Start X of object bounding box.*
- #define `L_start_ibb_y` 0.2
- Start Y of object bounding box.*
- #define `L_centre_ibb_z` 0.5
- Centre of object bounding box in Z direction.*
- #define `L_ibb_length` 0.5
- The object input is scaled based on this dimension.*
- #define `L_ibb_scale_direction` `eXDirection`
- Scale in this direction (specify as enumeration)*
- #define `L_ibb_length_ref` 0.5
- Reference length to be used in the definition of Reynolds number.*
- #define `L_num_markers` 124
- Number of Lagrange points to use when building a prefab body (approximately)*
- #define `L_ibb_deform` false
- Default deformable property of body to be built (whether it moves or not)*
- #define `L_ibb_flex_rigid` false
- Whether a structural calculation needs to be performed on the body.*
- #define `L_INSERT_CIRCLE_SPHERE`
- #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` 10.0
- Width (x) of IB body.*
- #define `L_ibb_l` 10.0
- Length (y) of IB body.*
- #define `L_ibb_d` 0.0
- Depth (z) of IB body.*
- #define `L_ibb_r` 0.05

- Radius of IB body.*
- #define [L_ibt_filament_length](#) 0.2
 - Length of filament.*
- #define [L_ibt_filament_start_x](#) 0.3
 - Start X position of the filament.*
- #define [L_ibt_filament_start_y](#) 0.0
 - Start Y position of the filament.*
- #define [L_ibt_filament_start_z](#) 0.0
 - Start Z position of the filament.*
- #define [L_ibt_angle_vert](#) 90
 - Inclination of filament in XY plane.*
- #define [L_ibt_angle_horz](#) 0
 - Inclination of filament in XZ plane.*
- #define [L_start_BC](#) 2
 - Type of boundary condition at filament start: 0 == free; 1 = simply supported; 2 == clamped.*
- #define [L_end_BC](#) 0
 - Type of boundary condition at filament end: 0 == free; 1 = simply supported; 2 == clamped.*
- #define [L_ibt_delta_rho](#) 1.0
 - Difference in density (lattice units) between solid and fluid.*
- #define [L_ibt_EI](#) 2.0
 - Flexural rigidity (lattice units) of filament.*
- #define [L_INLET_ON](#)
 - Turn on inlet boundary (assumed left-hand wall - default Do Nothing)*
- #define [L_INLET_REGULARISED](#)
 - Specify the inlet to be a regularised inlet condition (Latt & Chopard)*
- #define [L_OUTLET_ON](#)
 - Turn on outlet boundary (assumed right-hand wall – default First Order Extrap.)*
- #define [L_WALLS_ON](#)
 - Turn on no-slip walls (default is top, bottom, front, back unless L_WALLS_ON_2D is used)*
- #define [L_WALLS_ON_2D](#)
 - Limit no-slip walls to top and bottom no-slip walls only.*
- #define [L_wall_thickness_bottom](#) 1
 - Thickness of walls in coarsest lattice units.*
- #define [L_wall_thickness_top](#) 1
 - Thickness of top walls in coarsest lattice units.*
- #define [L_wall_thickness_front](#) 1
 - Thickness of front (3D) walls in coarsest lattice units.*
- #define [L_wall_thickness_back](#) 1
 - Thickness of back (3D) walls in coarsest lattice units.*
- #define [L_block_on_grid_lev](#) 0
 - Provide grid level on which block should be added.*
- #define [L_block_on_grid_reg](#) 0
 - Provide grid region on which block should be added.*
- #define [L_block_x_min](#) 32
 - Index of start of object/wall in x-direction.*
- #define [L_block_x_max](#) 64
 - Index of end of object/wall in x-direction.*
- #define [L_block_y_min](#) 16
 - Index of start of object/wall in y-direction.*
- #define [L_block_y_max](#) 48
 - Index of end of object/wall in y-direction.*

- `#define L_block_z_min 16`
Index of start of object/wall in z-direction.
- `#define L_block_z_max 48`
Index of end of object/wall in z-direction.
- `#define L_object_on_grid_lev 4`
Provide grid level on which object should be added.
- `#define L_object_on_grid_reg 0`
Provide grid region on which object should be added.
- `#define L_start_object_x 40`
Index for start of object bounding box in X direction.
- `#define L_start_object_y 16`
Index for start of object bounding box in Y direction.
- `#define L_centre_object_z 121`
Index for cetnre of object bounding box in Z direction.
- `#define L_object_length 160`
The object input is scaled based on this dimension.
- `#define L_object_scale_direction eXDirection`
Scale in this direction (specify as enumeration)
- `#define L_object_length_ref 160`
Reference length to be used in the definition of Reynolds number.
- `#define L_bfl_on_grid_lev 1`
Provide grid level on which BFL body should be added.
- `#define L_bfl_on_grid_reg 0`
Provide grid region on which BFL body should be added.
- `#define L_start_bfl_x 50`
Index for start of object bounding box in X direction.
- `#define L_start_bfl_y 100`
Index for start of object bounding box in Y direction.
- `#define L_centre_bfl_z 20`
Index for cetnre of object bounding box in Z direction.
- `#define L_bfl_length 50`
The BFL object input is scaled based on this dimension.
- `#define L_bfl_scale_direction eXDirection`
Scale in this direction (specify as enumeration)
- `#define L_bfl_length_ref 10`
Reference length to be used in the definition of Reynolds number.
- `#define L_NumLev 2`
Levels of refinement (0 = coarse grid only).
- `#define L_NumReg 1`
Number of refined regions (can be arbitrary if L_NumLev = 0)
- `#define L_nVels 9`
- `#define L_MPI_dir 8`
- `#define L_a_z 0`
Start of domain-z.
- `#define L_b_z 2`
End of domain-z.
- `#define L_K 1`
Number of z lattice sites.
- `#define L_block_z_min 0`
Index of start of object/wall in z-direction.
- `#define L_block_z_max 0`

- *Index of end of object/wall in z-direction.*
- `#define L_abb_d 0`
- *Depth (z) of IB body.*
- `#define L_centre_object_z 0`
- *Index for centre of object bounding box in Z direction.*
- `#define L_centre_bfl_z 0`
- *Index for centre of object bounding box in Z direction.*
- `#define L_centre_abb_z 0`
- *Centre of object bounding box in Z direction.*
- `#define L_u_0z 0`
- *Initial/inlet z-velocity.*
- `#define L_u_0x ux_in[j]`
- *Initial/inlet x-velocity.*
- `#define L_u_0y uy_in[j]`
- *Initial/inlet y-velocity.*
- `#define L_u_0z uz_in[j]`
- *Initial/inlet z-velocity.*

Variables

- static const int `nProbes` [3] = {3, 3, 3}
- *Number of probes in each direction (x, y, z)*
- static const int `xProbeLims` [2] = {90, 270}
- *Limits of X plane for array of probes.*
- static const int `yProbeLims` [2] = {15, 45}
- *Limits of Y plane for array of probes.*
- static const int `zProbeLims` [2] = {30, 120}
- *Limits of Z plane for array of probes.*
- static const int `RefXstart` [L_NumLev][L_NumReg] = { {5}, {5} }
- static const int `RefXend` [L_NumLev][L_NumReg] = { {110}, {150} }
- static const int `RefYstart` [L_NumLev][L_NumReg] = { {4}, {4} }
- static const int `RefYend` [L_NumLev][L_NumReg] = { {38}, {61} }
- static int `RefZstart` [L_NumLev][L_NumReg] = { {20}, {5} }
- static int `RefZend` [L_NumLev][L_NumReg] = { {40}, {35} }

6.6.1 Macro Definition Documentation

6.6.1.1 `#define L_a_x -0.005`

Start of domain-x.

6.6.1.2 `#define L_a_y -0.01`

Start of domain-y.

6.6.1.3 `#define L_a_z 0`

Start of domain-z.

6.6.1.4 `#define L_a_z 0`

Start of domain-z.

6.6.1.5 `#define L_b_x 2.205`

End of domain-x.

6.6.1.6 `#define L_b_y 0.42`

End of domain-y.

6.6.1.7 `#define L_b_z 6`

End of domain-z.

6.6.1.8 `#define L_b_z 2`

End of domain-z.

6.6.1.9 `#define L_bfl_length 50`

The BFL object input is scaled based on this dimension.

6.6.1.10 `#define L_bfl_length_ref 10`

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

6.6.1.11 `#define L_bfl_on_grid_lev 1`

Provide grid level on which BFL body should be added.

6.6.1.12 `#define L_bfl_on_grid_reg 0`

Provide grid region on which BFL body should be added.

6.6.1.13 `#define L_bfl_scale_direction eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.14 #define L_block_on_grid_lev 0

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_block_x_max 64

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

6.6.1.17 #define L_block_x_min 32

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

6.6.1.18 #define L_block_y_max 48

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

6.6.1.19 #define L_block_y_min 16

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

6.6.1.20 #define L_block_z_max 48

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

6.6.1.21 #define L_block_z_max 0

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

6.6.1.22 #define L_block_z_min 16

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

6.6.1.23 #define L_block_z_min 0

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

6.6.1.24 `#define L_centre_bfl_z 20`

Index for centre of object bounding box in Z direction.

6.6.1.25 `#define L_centre_bfl_z 0`

Index for centre of object bounding box in Z direction.

6.6.1.26 `#define L_centre_ibb_z 0.5`

Centre of object bounding box in Z direction.

6.6.1.27 `#define L_centre_ibb_z 0`

Centre of object bounding box in Z direction.

6.6.1.28 `#define L_centre_object_z 121`

Index for centre of object bounding box in Z direction.

6.6.1.29 `#define L_centre_object_z 0`

Index for centre of object bounding box in Z direction.

6.6.1.30 `#define L_dims 2`

Number of dimensions to the problem.

6.6.1.31 `#define L_end_BC 0`

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

6.6.1.32 `#define L_grav_direction eXDirection`

Gravity direction (specify using enumeration)

6.6.1.33 `#define L_grav_force 0`

Expression for the gravity force.

6.6.1.34 #define L_HDF5_OUTPUT

HDF5 dump on output.

6.6.1.35 #define L_IB_Lev 2

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

6.6.1.36 #define L_IB_Reg 0

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

6.6.1.37 #define L_ibb_angle_horz 0

Inclination of filament in XZ plane.

6.6.1.38 #define L_ibb_angle_vert 90

Inclination of filament in XY plane.

6.6.1.39 #define L_ibb_d 0.0

Depth (z) of IB body.

6.6.1.40 #define L_ibb_d 0

Depth (z) of IB body.

6.6.1.41 #define L_ibb_deform false

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

6.6.1.42 #define L_ibb_delta_rho 1.0

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

6.6.1.43 #define L_ibb_EI 2.0

Flexural rigidity (lattice units) of filament.

6.6.1.44 `#define L_abb_filament_length 0.2`

Length of filament.

6.6.1.45 `#define L_abb_filament_start_x 0.3`

Start X position of the filament.

6.6.1.46 `#define L_abb_filament_start_y 0.0`

Start Y position of the filament.

6.6.1.47 `#define L_abb_filament_start_z 0.0`

Start Z position of the filament.

6.6.1.48 `#define L_abb_flex_rigid false`

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

6.6.1.49 `#define L_abb_l 10.0`

Length (y) of IB body.

6.6.1.50 `#define L_abb_length 0.5`

The object input is scaled based on this dimension.

6.6.1.51 `#define L_abb_length_ref 0.5`

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

6.6.1.52 `#define L_abb_on_grid_lev L_IB_Lev`

Provide grid level on which object should be added.

6.6.1.53 `#define L_abb_on_grid_reg L_IB_Reg`

Provide grid region on which object should be added.

6.6.1.54 `#define L_abb_r 0.05`

Radius of IB body.

6.6.1.55 `#define L_abb_scale_direction eXDirection`

Scale in this direction (specify as enumeration)

6.6.1.56 `#define L_abb_w 10.0`

Width (x) of IB body.

6.6.1.57 `#define L_abb_x 0.2`

X Position of body centre.

6.6.1.58 `#define L_abb_y 0.2`

Y Position of body centre.

6.6.1.59 `#define L_abb_z 0.0`

Z Position of body centre.

6.6.1.60 `#define L_IBM_ON`

Turn on IBM.

6.6.1.61 `#define L_INLET_ON`

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

6.6.1.62 `#define L_INLET_REGULARISED`

Specify the inlet to be a regularised inlet condition (Latt & Chopard)

6.6.1.63 `#define L_INSERT_CIRCLE_SPHERE`

6.6.1.64 `#define L_K 60`

Number of z lattice sites.

6.6.1.65 `#define L_K 1`

Number of z lattice sites.

6.6.1.66 `#define L_M (1*41+2)`

Number of y lattice sites.

6.6.1.67 `#define L_MPI_dir 8`

6.6.1.68 `#define L_N (1*220+1)`

Number of x lattice sites.

6.6.1.69 `#define L_NO_FLOW`

Initialise the domain with no flow.

6.6.1.70 `#define L_num_markers 124`

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

6.6.1.71 `#define L_NumLev 2`

Levels of refinement (0 = coarse grid only).

6.6.1.72 `#define L_NumReg 1`

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

6.6.1.73 `#define L_nVels 9`

6.6.1.74 `#define L_object_length 160`

The object input is scaled based on this dimension.

6.6.1.75 `#define L_object_length_ref 160`

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

6.6.1.76 #define L_object_on_grid_lev 4

Provide grid level on which object should be added.

6.6.1.77 #define L_object_on_grid_reg 0

Provide grid region on which object should be added.

6.6.1.78 #define L_object_scale_direction eXDirection

Scale in this direction (specify as enumeration)

6.6.1.79 #define L_out_every 50

How many timesteps before whole grid output.

6.6.1.80 #define L_out_every_forces 50

Specific output frequency of body forces.

6.6.1.81 #define L_out_every_probe 250

Write out frequency of probe output.

6.6.1.82 #define L_OUTLET_ON

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

6.6.1.83 #define L_output_precision 6

Precision of output (for text writers)

6.6.1.84 #define L_PARABOLIC_INLET

Use analytic expression for inlet profile - if not then ASCII file is read (requires L_USE_INLET_PROFILE)

6.6.1.85 #define L_PI 3.14159265358979323846

PI definition.

6.6.1.86 `#define L_Re 20`

Desired Reynolds number.

6.6.1.87 `#define L_restart_out_every 10000`

Frequency of write out of restart file.

6.6.1.88 `#define L_rho_in 1`

Initial density.

6.6.1.89 `#define L_start_BC 2`

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

6.6.1.90 `#define L_start_bfl_x 50`

Index for start of object bounding box in X direction.

6.6.1.91 `#define L_start_bfl_y 100`

Index for start of object bounding box in Y direction.

6.6.1.92 `#define L_start_ibb_x 0.3`

Start X of object bounding box.

6.6.1.93 `#define L_start_ibb_y 0.2`

Start Y of object bounding box.

6.6.1.94 `#define L_start_object_x 40`

Index for start of object bounding box in X direction.

6.6.1.95 `#define L_start_object_y 16`

Index for start of object bounding box in Y direction.

6.6.1.96 `#define L_Timesteps 19800`

Number of time steps to run simulation for.

6.6.1.97 `#define L_u_0x L_u_ref`

Initial/inlet x-velocity.

6.6.1.98 `#define L_u_0x ux_in[j]`

Initial/inlet x-velocity.

6.6.1.99 `#define L_u_0y 0`

Initial/inlet y-velocity.

6.6.1.100 `#define L_u_0y uy_in[j]`

Initial/inlet y-velocity.

6.6.1.101 `#define L_u_0z 0`

Initial/inlet z-velocity.

6.6.1.102 `#define L_u_0z 0`

Initial/inlet z-velocity.

6.6.1.103 `#define L_u_0z uz_in[j]`

Initial/inlet z-velocity.

6.6.1.104 `#define L_u_max L_u_ref*1.5`

Max velocity of inlet profile.

6.6.1.105 `#define L_u_ref (0.0707070707070707)`

Reference velocity for scaling, can be mean inelt velocity.

6.6.1.106 #define L_USE_INLET_PROFILE

Use an inlet profile.

6.6.1.107 #define L_VTK_BODY_WRITE

Write out the bodies to a VTK file.

6.6.1.108 #define L_wall_thickness_back 1

Thickness of back (3D) walls in coarsest lattice units.

6.6.1.109 #define L_wall_thickness_bottom 1

Thickness of walls in coarsest lattice units.

6.6.1.110 #define L_wall_thickness_front 1

Thickness of front (3D) walls in coarsest lattice units.

6.6.1.111 #define L_wall_thickness_top 1

Thickness of top walls in coarsest lattice units.

6.6.1.112 #define L_WALLS_ON

Turn on no-slip walls (default is top, bottom, front, back unless L_WALLS_ON_2D is used)

6.6.1.113 #define L_WALLS_ON_2D

Limit no-slip walls to top and bottom no-slip walls only.

6.6.1.114 #define L_Xcores 2

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

6.6.1.115 #define L_Ycores 3

Number of MPI ranks to divide domain into in Y direction

6.6.1.116 `#define L_Zcores 2`

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

6.6.1.117 `#define LUMA_VERSION "1.2.0-alpha"`

LUMA version.

6.6.2 Variable Documentation

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

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

6.6.2.2 `const int RefXend[L_NumLev][L_NumReg] = { {110}, {150} } [static]`

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

6.6.2.4 `const int RefYend[L_NumLev][L_NumReg] = { {38}, {61} } [static]`

6.6.2.5 `const int RefYstart[L_NumLev][L_NumReg] = { {4}, {4} } [static]`

6.6.2.6 `int RefZend[L_NumLev][L_NumReg] = { {40}, {35} } [static]`

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

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

Limits of X plane for array of probes.

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

Limits of Y plane for array of probes.

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

Limits of Z plane for array of probes.

6.7 GridObj.cpp File Reference

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

6.8 GridObj.h File Reference

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

Classes

- class [GridObj](#)
Grid class.

Enumerations

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

6.8.1 Enumeration Type Documentation

6.8.1.1 enum [eBCType](#)

Flag for indicating which BCs to apply.

Enumerator

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

6.8.1.2 enum eType

Lattice typing labels.

Enumerator

eSolid Rigid, solid site.

eFluid Fluid site.

eRefined Fluid site which is represented on a finer grid.

eTransitionToCoarser Fluid site coupled to a coarser grid.

eTransitionToFiner Fluid site coupled to a finer grid.

eBFL Site containing a BFL marker.

eSymmetry Symmetry boundary.

eInlet Inlet boundary.

eOutlet Outlet boundary.

eRefinedSolid Rigid, solid site represented on a finer grid.

eRefinedSymmetry Symmetry boundary represented on a finer grid.

eRefinedInlet Inlet site represented on a finer grid.

6.9 GridObj_init_grids.cpp File Reference

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

6.10 GridObj_ops_boundary.cpp File Reference

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

6.11 GridObj_ops_io.cpp File Reference

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

6.12 GridObj_ops_lbm.cpp File Reference

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

6.13 GridUtils.cpp File Reference

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

6.14 GridUtils.h File Reference

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

Classes

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

Enumerations

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

6.14.1 Enumeration Type Documentation

6.14.1.1 enum [eCartesianDirection](#)

Enumeration for directional options.

Enumerator

[eXDirection](#) X-direction.
[eYDirection](#) Y-direction.
[eZDirection](#) Z-direction.

6.14.1.2 enum eMinMax

Enumeration for minimum and maximum.

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

Enumerator

eMinimum Minimum.

eMaximum Maximum.

6.15 hdf5luma.h File Reference

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

Macros

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

Enumerations

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

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

Functions

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

Helper method to write out using HDF5.

6.15.1 Macro Definition Documentation

6.15.1.1 `#define H5_BUILT_AS_DYNAMIC_LIB`

6.15.1.2 `#define HDF5_EXT_SZIP`

6.15.1.3 `#define HDF5_EXT_ZLIB`

6.15.2 Enumeration Type Documentation

6.15.2.1 `enum eHdf5SlabType`

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

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

Enumerator

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

6.15.3 Function Documentation

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

Helper method to write out using HDF5.

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

Parameters

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

6.16 IBody.cpp File Reference

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

6.17 IBody.h File Reference

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

Classes

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

6.18 IBMarker.cpp File Reference

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

6.19 IBMarker.h File Reference

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

Classes

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

6.20 IVector.h File Reference

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

Classes

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

6.21 main_lbm.cpp File Reference

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

Functions

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

6.21.1 Function Documentation

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

Entry point for the application.

6.22 Marker.h File Reference

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

Classes

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

6.23 Mpi_buffer_pack.cpp File Reference

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


6.24 `Mpi_buffer_size_rcv.cpp` File Reference

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

6.25 `Mpi_buffer_size_send.cpp` File Reference

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

6.26 `Mpi_buffer_unpk.cpp` File Reference

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

6.27 `MpiManager.cpp` File Reference

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

6.28 `MpiManager.h` File Reference

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

Classes

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

Macros

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

6.28.1 Macro Definition Documentation

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

For loop definition for left halo.

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

For loop definition for right halo.

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

For loop definition for bottom halo.

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

For loop definition for top halo.

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

For loop definition for back halo.

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

For loop definition for front halo.

6.29 ObjectManager.cpp File Reference

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

6.30 ObjectManager.h File Reference

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

Classes

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

Enumerations

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

6.30.1 Enumeration Type Documentation

6.30.1.1 enum eObjectType

Specifies the type of body being processed.

Enumerator

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

6.31 ObjectManager_init_bflbody.cpp File Reference

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

6.32 ObjectManager_init_ibmbody.cpp File Reference

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

6.33 ObjectManager_ops_ibm.cpp File Reference

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

6.34 ObjectManager_ops_ibmflex.cpp File Reference

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

Macros

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

6.34.1 Macro Definition Documentation

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

Pointer swap definition.

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

Pointer swap definition.

6.34.1.3 `#define TINY 1.0e-20`

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

6.35 ObjectManager_ops_io.cpp File Reference

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

6.36 PCpts.h File Reference

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

Classes

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

6.37 stdafx.cpp File Reference

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

Variables

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

6.37.1 Variable Documentation

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

Initial value:

```
=
{
  { 1, -1, 0, 0, 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.

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

Lattice sound speed.

6.37.1.3 `const double w[L_nVels]`

Initial value:

```
=
{ 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.

6.38 `stdafx.h` File Reference

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

Macros

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

Variables

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

6.38.1 Macro Definition Documentation

6.38.1.1 `#define L_IS_NAN std::isnan`

Not a Number declaration (Unix)

6.38.1.2 `#define LUMA_FAILED 12345`

Error definition.

6.38.2 Variable Documentation

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

Lattice velocities.

6.38.2.2 `const double cs`

Lattice sound speed.

6.38.2.3 `const double w[L_nVels]`

Quadrature weights.

Index

- _Owner
 - Body, [16](#)
 - ~BFLBody
 - BFLBody, [10](#)
 - ~BFLMarker
 - BFLMarker, [12](#)
 - ~Body
 - Body, [14](#)
 - ~GridObj
 - GridObj, [21](#)
 - ~IBBody
 - IBBody, [46](#)
 - ~IBMarker
 - IBMarker, [50](#)
 - ~IVector
 - IVector, [53](#)
 - ~Marker
 - Marker, [56](#)
 - ~MarkerData
 - MarkerData, [58](#)
 - ~PCpts
 - PCpts, [77](#)
- add
 - GridUtils, [32](#)
- addMarker
 - Body, [14](#)
 - IBBody, [47](#)
- BCs
 - IBBody, [49](#)
- BFLBody, [9](#)
 - ~BFLBody, [10](#)
 - BFLBody, [10](#)
 - BFLMarker, [12](#)
 - computeQ, [10](#), [11](#)
 - GridObj, [11](#)
 - Q, [11](#)
- BFLBody.cpp, [81](#)
- BFLBody.h, [81](#)
- BFLMarker, [11](#)
 - ~BFLMarker, [12](#)
 - BFLBody, [12](#)
 - BFLMarker, [12](#)
- BFLMarker.cpp, [81](#)
- BFLMarker.h, [82](#)
- bc_applyBfl
 - GridObj, [21](#)
- bc_applyBounceBack
 - GridObj, [21](#)
- bc_applyExtrapolation
 - GridObj, [21](#)
- bc_applyNrbc
 - GridObj, [22](#)
- bc_applyRegularised
 - GridObj, [22](#)
- bc_applySpecReflect
 - GridObj, [22](#)
- bc_solidSiteReset
 - GridObj, [22](#)
- bfl_build_body
 - ObjectManager, [69](#)
- Body
 - _Owner, [16](#)
 - ~Body, [14](#)
 - addMarker, [14](#)
 - Body, [14](#)
 - closed_surface, [16](#)
 - getMarkerData, [14](#)
 - isInVoxel, [15](#)
 - isVoxelMarkerVoxel, [15](#)
 - markerAdder, [15](#)
 - markers, [16](#)
 - spacing, [16](#)
- Body< MarkerType >, [13](#)
- Body.h, [82](#)
- buffer_rcv_info
 - MpiManager, [64](#)
- buffer_send_info
 - MpiManager, [64](#)
- c
 - stdafx.cpp, [112](#)
 - stdafx.h, [113](#)
- closed_surface
 - Body, [16](#)
- computeLiftDrag
 - ObjectManager, [69](#)
- computeQ
 - BFLBody, [10](#), [11](#)
- createOutputDirectory
 - GridUtils, [32](#)
- crossprod
 - GridUtils, [33](#)
- cs
 - stdafx.cpp, [112](#)
 - stdafx.h, [113](#)
- definitions.h, [82](#)
 - L_HDF5_OUTPUT, [90](#)

[L_IB_Lev, 91](#)
[L_IB_Reg, 91](#)
[L_IBM_ON, 93](#)
[L_INLET_ON, 93](#)
[L_INLET_REGULARISED, 93](#)
[L_INSERT_CIRCLE_SPHERE, 93](#)
[L_MPI_dir, 94](#)
[L_NO_FLOW, 94](#)
[L_NumLev, 94](#)
[L_NumReg, 94](#)
[L_OUTLET_ON, 95](#)
[L_PARABOLIC_INLET, 95](#)
[L_PI, 95](#)
[L_Re, 95](#)
[L_Timesteps, 96](#)
[L_USE_INLET_PROFILE, 97](#)
[L_VTK_BODY_WRITE, 98](#)
[L_WALLS_ON_2D, 98](#)
[L_WALLS_ON, 98](#)
[L_Xcores, 98](#)
[L_Ycores, 98](#)
[L_Zcores, 98](#)
[L_a_x, 87](#)
[L_a_y, 87](#)
[L_a_z, 87](#)
[L_b_x, 88](#)
[L_b_y, 88](#)
[L_b_z, 88](#)
[L_bfl_length, 88](#)
[L_bfl_length_ref, 88](#)
[L_bfl_on_grid_lev, 88](#)
[L_bfl_on_grid_reg, 88](#)
[L_bfl_scale_direction, 88](#)
[L_block_on_grid_lev, 88](#)
[L_block_on_grid_reg, 89](#)
[L_block_x_max, 89](#)
[L_block_x_min, 89](#)
[L_block_y_max, 89](#)
[L_block_y_min, 89](#)
[L_block_z_max, 89](#)
[L_block_z_min, 89](#)
[L_centre_bfl_z, 89, 90](#)
[L_centre_ibb_z, 90](#)
[L_centre_object_z, 90](#)
[L_dims, 90](#)
[L_end_BC, 90](#)
[L_grav_direction, 90](#)
[L_grav_force, 90](#)
[L_ibb_EI, 91](#)
[L_ibb_angle_horz, 91](#)
[L_ibb_angle_vert, 91](#)
[L_ibb_d, 91](#)
[L_ibb_deform, 91](#)
[L_ibb_delta_rho, 91](#)
[L_ibb_filament_length, 91](#)
[L_ibb_filament_start_x, 92](#)
[L_ibb_filament_start_y, 92](#)
[L_ibb_filament_start_z, 92](#)
[L_ibb_flex_rigid, 92](#)
[L_ibb_I, 92](#)
[L_ibb_length, 92](#)
[L_ibb_length_ref, 92](#)
[L_ibb_on_grid_lev, 92](#)
[L_ibb_on_grid_reg, 92](#)
[L_ibb_r, 92](#)
[L_ibb_scale_direction, 93](#)
[L_ibb_w, 93](#)
[L_ibb_x, 93](#)
[L_ibb_y, 93](#)
[L_ibb_z, 93](#)
[L_K, 93](#)
[L_M, 94](#)
[L_N, 94](#)
[L_nVels, 94](#)
[L_num_markers, 94](#)
[L_object_length, 94](#)
[L_object_length_ref, 94](#)
[L_object_on_grid_lev, 94](#)
[L_object_on_grid_reg, 95](#)
[L_object_scale_direction, 95](#)
[L_out_every, 95](#)
[L_out_every_forces, 95](#)
[L_out_every_probe, 95](#)
[L_output_precision, 95](#)
[L_restart_out_every, 96](#)
[L_rho_in, 96](#)
[L_start_BC, 96](#)
[L_start_bfl_x, 96](#)
[L_start_bfl_y, 96](#)
[L_start_ibb_x, 96](#)
[L_start_ibb_y, 96](#)
[L_start_object_x, 96](#)
[L_start_object_y, 96](#)
[L_u_0x, 97](#)
[L_u_0y, 97](#)
[L_u_0z, 97](#)
[L_u_max, 97](#)
[L_u_ref, 97](#)
[L_wall_thickness_back, 98](#)
[L_wall_thickness_bottom, 98](#)
[L_wall_thickness_front, 98](#)
[L_wall_thickness_top, 98](#)
[LUMA_VERSION, 99](#)
[nProbes, 99](#)
[RefXend, 99](#)
[RefXstart, 99](#)
[RefYend, 99](#)
[RefYstart, 99](#)
[RefZend, 99](#)
[RefZstart, 99](#)
[xProbeLims, 99](#)
[yProbeLims, 99](#)
[zProbeLims, 99](#)
[deformable](#)
[IBBody, 49](#)
[delta_rho](#)

- IBBody, 49
- deltaval
 - IBMarker, 51
- desired_vel
 - IBMarker, 51
- destroyInstance
 - MpiManager, 62
 - ObjectManager, 70
- dilation
 - IBMarker, 51
- dir_reflect
 - GridUtils, 44
- dotprod
 - GridUtils, 33
- downToLimit
 - GridUtils, 33
- dt
 - GridObj, 28
- eBBBCloud
 - ObjectManager.h, 109
- eBCAll
 - GridObj.h, 100
- eBCBFL
 - GridObj.h, 100
- eBCInlet
 - GridObj.h, 100
- eBCInletOutlet
 - GridObj.h, 100
- eBCOutlet
 - GridObj.h, 100
- eBCSolidSymmetry
 - GridObj.h, 100
- eBCType
 - GridObj.h, 100
- eBFLCloud
 - ObjectManager.h, 109
- eBFL
 - GridObj.h, 101
- eCartesianDirection
 - GridUtils.h, 102
- eFluid
 - GridObj.h, 101
- eHdf5SlabType
 - hdf5luma.h, 104
- eIBBCloud
 - ObjectManager.h, 109
- eInlet
 - GridObj.h, 101
- eMaximum
 - GridUtils.h, 103
- eMinMax
 - GridUtils.h, 102
- eMinimum
 - GridUtils.h, 103
- eObjectType
 - ObjectManager.h, 109
- eOutlet
 - GridObj.h, 101
- ePosX
 - hdf5luma.h, 104
- ePosY
 - hdf5luma.h, 104
- ePosZ
 - hdf5luma.h, 104
- eProductVector
 - hdf5luma.h, 104
- eRefined
 - GridObj.h, 101
- eRefinedInlet
 - GridObj.h, 101
- eRefinedSolid
 - GridObj.h, 101
- eRefinedSymmetry
 - GridObj.h, 101
- eScalar
 - hdf5luma.h, 104
- eSolid
 - GridObj.h, 101
- eSymmetry
 - GridObj.h, 101
- eTransitionToCoarser
 - GridObj.h, 101
- eTransitionToFiner
 - GridObj.h, 101
- eType
 - GridObj.h, 100
- eVector
 - hdf5luma.h, 104
- eXDirection
 - GridUtils.h, 102
- eYDirection
 - GridUtils.h, 102
- eZDirection
 - GridUtils.h, 102
- epsilon
 - IBMarker, 51
- f_buffer_recv
 - MpiManager, 64
- f_buffer_send
 - MpiManager, 64
- factorial
 - GridUtils, 34
- flex_rigid
 - IBBody, 49
 - IBMarker, 51
- flexural_rigidity
 - IBBody, 49
- fluid_vel
 - IBMarker, 51
- force_xyz
 - IBMarker, 52
- getCoarseIndices
 - GridUtils, 34
- getFineIndices
 - GridUtils, 34

- getGrid
 - GridUtils, 35
- getInstance
 - MpiManager, 62
 - ObjectManager, 70
- getMarkerData
 - Body, 14
- getOpposite
 - GridUtils, 35
- getVoxInd
 - GridUtils, 36
- global_dims
 - MpiManager, 65
- global_edge_ind
 - MpiManager, 65
- global_edge_pos
 - MpiManager, 65
- global_to_local
 - GridUtils, 36
- GridObj, 17
 - ~GridObj, 21
 - BFLBody, 11
 - bc_applyBfl, 21
 - bc_applyBounceBack, 21
 - bc_applyExtrapolation, 21
 - bc_applyNrbc, 22
 - bc_applyRegularised, 22
 - bc_applySpecReflect, 22
 - bc_solidSiteReset, 22
 - dt, 28
 - GridObj, 20
 - GridUtils, 28
 - io_hdf5, 23
 - io_lite, 23
 - io_probeOutput, 23
 - io_restart, 23
 - io_textout, 23
 - K_lim, 28
 - LBM_addSubGrid, 24
 - LBM_boundary, 24
 - LBM_coalesce, 24
 - LBM_collide, 24
 - LBM_explode, 25
 - LBM_forceGrid, 25
 - LBM_init_getInletProfile, 25
 - LBM_initBoundLab, 25
 - LBM_initGrid, 25, 26
 - LBM_initRefinedLab, 26
 - LBM_initRho, 26
 - LBM_initSolidLab, 26
 - LBM_initSubGrid, 26
 - LBM_initVelocity, 27
 - LBM_kbcCollide, 27
 - LBM_macro, 27
 - LBM_multi, 27
 - LBM_resetForces, 28
 - LBM_stream, 28
 - LatTyp, 28
 - level, 28
 - M_lim, 28
 - MpiManager, 28
 - N_lim, 29
 - nu, 29
 - ObjectManager, 28, 76
 - omega, 29
 - t, 29
 - timeav_mpi_overhead, 29
 - timeav_timestep, 29
 - XInd, 29
 - XPos, 29
 - YInd, 29
 - YPos, 29
 - ZInd, 30
 - ZPos, 30
- GridObj.cpp, 100
- GridObj.h, 100
 - eBCAll, 100
 - eBCBFL, 100
 - eBCInlet, 100
 - eBCInletOutlet, 100
 - eBCOutlet, 100
 - eBCSolidSymmetry, 100
 - eBCType, 100
 - eBFL, 101
 - eFluid, 101
 - eInlet, 101
 - eOutlet, 101
 - eRefined, 101
 - eRefinedInlet, 101
 - eRefinedSolid, 101
 - eRefinedSymmetry, 101
 - eSolid, 101
 - eSymmetry, 101
 - eTransitionToCoarser, 101
 - eTransitionToFiner, 101
 - eType, 100
- GridObj_init_grids.cpp, 101
- GridObj_ops_boundary.cpp, 101
- GridObj_ops_io.cpp, 101
- GridObj_ops_lbm.cpp, 102
- GridUtils, 30
 - add, 32
 - createOutputDirectory, 32
 - crossprod, 33
 - dir_reflect, 44
 - dotprod, 33
 - downToLimit, 33
 - factorial, 34
 - getCoarseIndices, 34
 - getFineIndices, 34
 - getGrid, 35
 - getOpposite, 35
 - getVoxInd, 36
 - global_to_local, 36
 - GridObj, 28
 - hasThisSubGrid, 37

- isOffGrid, 37
- isOnRecvLayer, 37, 38
- isOnSenderLayer, 38
- isOnThisRank, 39
- isOverlapPeriodic, 39
- linspace, 40
- local_to_global, 40
- logfile, 44
- matrix_multiply, 41
- onespace, 41
- path_str, 45
- stridedCopy, 41
- subtract, 41
- upToZero, 42
- vecmultiply, 42
- vecnorm, 42–44
- GridUtils.cpp, 102
- GridUtils.h, 102
 - eCartesianDirection, 102
 - eMaximum, 103
 - eMinMax, 102
 - eMinimum, 103
 - eXDirection, 102
 - eYDirection, 102
 - eZDirection, 102
- Grids
 - MpiManager, 65
- groupID
 - IBBody, 49
- H5_BUILT_AS_DYNAMIC_LIB
 - hdf5luma.h, 104
- HDF5_EXT_SZIP
 - hdf5luma.h, 104
- HDF5_EXT_ZLIB
 - hdf5luma.h, 104
- halo_max
 - MpiManager::phdf5_struct, 78
- halo_min
 - MpiManager::phdf5_struct, 78
- hasThisSubGrid
 - GridUtils, 37
- hdf5_writeDataSet
 - hdf5luma.h, 104
- hdf5luma.h, 103
 - eHdf5SlabType, 104
 - ePosX, 104
 - ePosY, 104
 - ePosZ, 104
 - eProductVector, 104
 - eScalar, 104
 - eVector, 104
 - H5_BUILT_AS_DYNAMIC_LIB, 104
 - HDF5_EXT_SZIP, 104
 - HDF5_EXT_ZLIB, 104
 - hdf5_writeDataSet, 104
- i
 - MarkerData, 59
 - i_end
 - MpiManager::phdf5_struct, 78
 - i_start
 - MpiManager::phdf5_struct, 79
 - IBBody, 45
 - ~IBBody, 46
 - addMarker, 47
 - BCs, 49
 - deformable, 49
 - delta_rho, 49
 - flex_rigid, 49
 - flexural_rigidity, 49
 - groupID, 49
 - IBBody, 46
 - IBMarker, 51
 - makeBody, 47, 48
 - ObjectManager, 48
 - tension, 49
 - IBBody.cpp, 105
 - IBBody.h, 105
 - IBMarker, 49
 - ~IBMarker, 50
 - deltaval, 51
 - desired_vel, 51
 - dilation, 51
 - epsilon, 51
 - flex_rigid, 51
 - fluid_vel, 51
 - force_xyz, 52
 - IBBody, 51
 - IBMarker, 50, 51
 - local_area, 52
 - ObjectManager, 51
 - position_old, 52
 - IBMarker.cpp, 105
 - IBMarker.h, 105
 - IVector
 - ~IVector, 53
 - IVector, 53
 - operator(), 53, 54
 - IVector< GenTyp >, 52
 - IVector.h, 105
 - ibm_apply
 - ObjectManager, 70
 - ibm_banbks
 - ObjectManager, 70
 - ibm_banded
 - ObjectManager, 71
 - ibm_bicgstab
 - ObjectManager, 71
 - ibm_build_body
 - ObjectManager, 72
 - ibm_computeforce
 - ObjectManager, 72
 - ibm_deltakernel
 - ObjectManager, 72
 - ibm_findepilon
 - ObjectManager, 73

- ibm_findsupport
 - ObjectManager, [73](#)
- ibm_initialise
 - ObjectManager, [73](#)
- ibm_interpol
 - ObjectManager, [73](#)
- ibm_jacowire
 - ObjectManager, [73](#)
- ibm_move_bodies
 - ObjectManager, [74](#)
- ibm_position_update
 - ObjectManager, [74](#)
- ibm_position_update_grp
 - ObjectManager, [74](#)
- ibm_spread
 - ObjectManager, [74](#)
- ID
 - MarkerData, [59](#)
- io_hdf5
 - GridObj, [23](#)
- io_lite
 - GridObj, [23](#)
- io_probeOutput
 - GridObj, [23](#)
- io_readInCloud
 - ObjectManager, [75](#)
- io_restart
 - GridObj, [23](#)
 - ObjectManager, [75](#)
- io_textout
 - GridObj, [23](#)
- io_vtk_IBwriter
 - ObjectManager, [75](#)
- io_write_body_pos
 - ObjectManager, [75](#)
- io_write_lift_drag
 - ObjectManager, [75](#)
- io_writeForceOnObject
 - ObjectManager, [76](#)
- isInVoxel
 - Body, [15](#)
- isOffGrid
 - GridUtils, [37](#)
- isOnRecvLayer
 - GridUtils, [37](#), [38](#)
- isOnSenderLayer
 - GridUtils, [38](#)
- isOnThisRank
 - GridUtils, [39](#)
- isOverlapPeriodic
 - GridUtils, [39](#)
- isVoxelMarkerVoxel
 - Body, [15](#)
- j
 - MarkerData, [59](#)
- j_end
 - MpiManager::phdf5_struct, [79](#)
- j_start
 - MpiManager::phdf5_struct, [79](#)
- k
 - MarkerData, [59](#)
- k_end
 - MpiManager::phdf5_struct, [79](#)
- K_lim
 - GridObj, [28](#)
- k_start
 - MpiManager::phdf5_struct, [79](#)
- L_HDF5_OUTPUT
 - definitions.h, [90](#)
- L_IB_Lev
 - definitions.h, [91](#)
- L_IB_Reg
 - definitions.h, [91](#)
- L_IBM_ON
 - definitions.h, [93](#)
- L_INLET_ON
 - definitions.h, [93](#)
- L_INLET_REGULARISED
 - definitions.h, [93](#)
- L_INSERT_CIRCLE_SPHERE
 - definitions.h, [93](#)
- L_IS_NAN
 - stdafx.h, [113](#)
- L_MPI_dir
 - definitions.h, [94](#)
- L_NO_FLOW
 - definitions.h, [94](#)
- L_NumLev
 - definitions.h, [94](#)
- L_NumReg
 - definitions.h, [94](#)
- L_OUTLET_ON
 - definitions.h, [95](#)
- L_PARABOLIC_INLET
 - definitions.h, [95](#)
- L_PI
 - definitions.h, [95](#)
- L_Re
 - definitions.h, [95](#)
- L_Timesteps
 - definitions.h, [96](#)
- L_USE_INLET_PROFILE
 - definitions.h, [97](#)
- L_VTK_BODY_WRITE
 - definitions.h, [98](#)
- L_WALLS_ON_2D
 - definitions.h, [98](#)
- L_WALLS_ON
 - definitions.h, [98](#)
- L_Xcores
 - definitions.h, [98](#)
- L_Ycores
 - definitions.h, [98](#)
- L_Zcores
 - definitions.h, [98](#)

- L_a_x
 - definitions.h, [87](#)
- L_a_y
 - definitions.h, [87](#)
- L_a_z
 - definitions.h, [87](#)
- L_b_x
 - definitions.h, [88](#)
- L_b_y
 - definitions.h, [88](#)
- L_b_z
 - definitions.h, [88](#)
- L_bfl_length
 - definitions.h, [88](#)
- L_bfl_length_ref
 - definitions.h, [88](#)
- L_bfl_on_grid_lev
 - definitions.h, [88](#)
- L_bfl_on_grid_reg
 - definitions.h, [88](#)
- L_bfl_scale_direction
 - definitions.h, [88](#)
- L_block_on_grid_lev
 - definitions.h, [88](#)
- L_block_on_grid_reg
 - definitions.h, [89](#)
- L_block_x_max
 - definitions.h, [89](#)
- L_block_x_min
 - definitions.h, [89](#)
- L_block_y_max
 - definitions.h, [89](#)
- L_block_y_min
 - definitions.h, [89](#)
- L_block_z_max
 - definitions.h, [89](#)
- L_block_z_min
 - definitions.h, [89](#)
- L_centre_bfl_z
 - definitions.h, [89](#), [90](#)
- L_centre_ibt_z
 - definitions.h, [90](#)
- L_centre_object_z
 - definitions.h, [90](#)
- L_dims
 - definitions.h, [90](#)
- L_end_BC
 - definitions.h, [90](#)
- L_grav_direction
 - definitions.h, [90](#)
- L_grav_force
 - definitions.h, [90](#)
- L_ibt_EI
 - definitions.h, [91](#)
- L_ibt_angle_horz
 - definitions.h, [91](#)
- L_ibt_angle_vert
 - definitions.h, [91](#)
- L_ibt_d
 - definitions.h, [91](#)
- L_ibt_deform
 - definitions.h, [91](#)
- L_ibt_delta_rho
 - definitions.h, [91](#)
- L_ibt_filament_length
 - definitions.h, [91](#)
- L_ibt_filament_start_x
 - definitions.h, [92](#)
- L_ibt_filament_start_y
 - definitions.h, [92](#)
- L_ibt_filament_start_z
 - definitions.h, [92](#)
- L_ibt_flex_rigid
 - definitions.h, [92](#)
- L_ibt_l
 - definitions.h, [92](#)
- L_ibt_length
 - definitions.h, [92](#)
- L_ibt_length_ref
 - definitions.h, [92](#)
- L_ibt_on_grid_lev
 - definitions.h, [92](#)
- L_ibt_on_grid_reg
 - definitions.h, [92](#)
- L_ibt_r
 - definitions.h, [92](#)
- L_ibt_scale_direction
 - definitions.h, [93](#)
- L_ibt_w
 - definitions.h, [93](#)
- L_ibt_x
 - definitions.h, [93](#)
- L_ibt_y
 - definitions.h, [93](#)
- L_ibt_z
 - definitions.h, [93](#)
- L_K
 - definitions.h, [93](#)
- L_M
 - definitions.h, [94](#)
- L_N
 - definitions.h, [94](#)
- L_nVels
 - definitions.h, [94](#)
- L_num_markers
 - definitions.h, [94](#)
- L_object_length
 - definitions.h, [94](#)
- L_object_length_ref
 - definitions.h, [94](#)
- L_object_on_grid_lev
 - definitions.h, [94](#)
- L_object_on_grid_reg
 - definitions.h, [95](#)
- L_object_scale_direction
 - definitions.h, [95](#)

- L_out_every
 - definitions.h, [95](#)
- L_out_every_forces
 - definitions.h, [95](#)
- L_out_every_probe
 - definitions.h, [95](#)
- L_output_precision
 - definitions.h, [95](#)
- L_restart_out_every
 - definitions.h, [96](#)
- L_rho_in
 - definitions.h, [96](#)
- L_start_BC
 - definitions.h, [96](#)
- L_start_bfl_x
 - definitions.h, [96](#)
- L_start_bfl_y
 - definitions.h, [96](#)
- L_start_ibt_x
 - definitions.h, [96](#)
- L_start_ibt_y
 - definitions.h, [96](#)
- L_start_object_x
 - definitions.h, [96](#)
- L_start_object_y
 - definitions.h, [96](#)
- L_u_0x
 - definitions.h, [97](#)
- L_u_0y
 - definitions.h, [97](#)
- L_u_0z
 - definitions.h, [97](#)
- L_u_max
 - definitions.h, [97](#)
- L_u_ref
 - definitions.h, [97](#)
- L_wall_thickness_back
 - definitions.h, [98](#)
- L_wall_thickness_bottom
 - definitions.h, [98](#)
- L_wall_thickness_front
 - definitions.h, [98](#)
- L_wall_thickness_top
 - definitions.h, [98](#)
- LBM_addSubGrid
 - GridObj, [24](#)
- LBM_boundary
 - GridObj, [24](#)
- LBM_coalesce
 - GridObj, [24](#)
- LBM_collide
 - GridObj, [24](#)
- LBM_explode
 - GridObj, [25](#)
- LBM_forceGrid
 - GridObj, [25](#)
- LBM_init_getInletProfile
 - GridObj, [25](#)
- LBM_initBoundLab
 - GridObj, [25](#)
- LBM_initGrid
 - GridObj, [25](#), [26](#)
- LBM_initRefinedLab
 - GridObj, [26](#)
- LBM_initRho
 - GridObj, [26](#)
- LBM_initSolidLab
 - GridObj, [26](#)
- LBM_initSubGrid
 - GridObj, [26](#)
- LBM_initVelocity
 - GridObj, [27](#)
- LBM_kbcCollide
 - GridObj, [27](#)
- LBM_macro
 - GridObj, [27](#)
- LBM_multi
 - GridObj, [27](#)
- LBM_resetForces
 - GridObj, [28](#)
- LBM_stream
 - GridObj, [28](#)
- LUMA_FAILED
 - stdafx.h, [113](#)
- LUMA_VERSION
 - definitions.h, [99](#)
- LatTyp
 - GridObj, [28](#)
- level
 - GridObj, [28](#)
 - MpiManager::buffer_struct, [17](#)
 - MpiManager::phdf5_struct, [79](#)
- linspace
 - GridUtils, [40](#)
- local_area
 - IBMarker, [52](#)
- local_size
 - MpiManager, [65](#)
- local_to_global
 - GridUtils, [40](#)
- logfile
 - GridUtils, [44](#)
- logout
 - MpiManager, [65](#)
- M_lim
 - GridObj, [28](#)
- MPI_cartlab
 - MpiManager, [65](#)
- MPI_coords
 - MpiManager, [65](#)
- MPI_dims
 - MpiManager, [66](#)
- main
 - main_lbm.cpp, [106](#)
- main_lbm.cpp, [106](#)
- main, [106](#)

- makeBody
 - IBBody, [47](#), [48](#)
- Marker, [55](#)
 - ~Marker, [56](#)
 - Marker, [56](#)
 - position, [57](#)
 - supp_i, [57](#)
 - supp_j, [57](#)
 - supp_k, [57](#)
 - support_rank, [57](#)
- Marker.h, [106](#)
- markerAdder
 - Body, [15](#)
- MarkerData, [57](#)
 - ~MarkerData, [58](#)
 - i, [59](#)
 - ID, [59](#)
 - j, [59](#)
 - k, [59](#)
 - MarkerData, [58](#)
 - x, [59](#)
 - y, [59](#)
 - z, [59](#)
- markers
 - Body, [16](#)
- matrix_multiply
 - GridUtils, [41](#)
- mpi_buffer_pack
 - MpiManager, [62](#)
- Mpi_buffer_pack.cpp, [106](#)
- mpi_buffer_size
 - MpiManager, [62](#)
- mpi_buffer_size_recv
 - MpiManager, [62](#)
- Mpi_buffer_size_recv.cpp, [107](#)
- mpi_buffer_size_send
 - MpiManager, [63](#)
- Mpi_buffer_size_send.cpp, [107](#)
- mpi_buffer_unpack
 - MpiManager, [63](#)
- Mpi_buffer_unpk.cpp, [107](#)
- mpi_buildCommunicators
 - MpiManager, [63](#)
- mpi_communicate
 - MpiManager, [63](#)
- mpi_getOpposite
 - MpiManager, [64](#)
- mpi_gridbuild
 - MpiManager, [64](#)
- mpi_init
 - MpiManager, [64](#)
- mpi_writeout_buf
 - MpiManager, [64](#)
- MpiManager, [59](#)
 - buffer_recv_info, [64](#)
 - buffer_send_info, [64](#)
 - destroyInstance, [62](#)
 - f_buffer_recv, [64](#)
 - f_buffer_send, [64](#)
 - getInstance, [62](#)
 - global_dims, [65](#)
 - global_edge_ind, [65](#)
 - global_edge_pos, [65](#)
 - GridObj, [28](#)
 - Grids, [65](#)
 - local_size, [65](#)
 - logout, [65](#)
 - MPI_cartlab, [65](#)
 - MPI_coords, [65](#)
 - MPI_dims, [66](#)
 - mpi_buffer_pack, [62](#)
 - mpi_buffer_size, [62](#)
 - mpi_buffer_size_recv, [62](#)
 - mpi_buffer_size_send, [63](#)
 - mpi_buffer_unpack, [63](#)
 - mpi_buildCommunicators, [63](#)
 - mpi_communicate, [63](#)
 - mpi_getOpposite, [64](#)
 - mpi_gridbuild, [64](#)
 - mpi_init, [64](#)
 - mpi_writeout_buf, [64](#)
 - my_rank, [66](#)
 - neighbour_coords, [66](#)
 - neighbour_rank, [66](#)
 - num_ranks, [66](#)
 - p_data, [66](#)
 - recv_layer_pos, [66](#)
 - recv_stat, [66](#)
 - send_requests, [66](#)
 - send_stat, [66](#)
 - sender_layer_pos, [67](#)
 - subGrid_comm, [67](#)
 - world_comm, [67](#)
- MpiManager.cpp, [107](#)
- MpiManager.h, [107](#)
 - range_i_left, [108](#)
 - range_i_right, [108](#)
 - range_j_down, [108](#)
 - range_j_up, [108](#)
 - range_k_back, [108](#)
 - range_k_front, [109](#)
- MpiManager::buffer_struct, [16](#)
 - level, [17](#)
 - region, [17](#)
 - size, [17](#)
- MpiManager::layer_edges, [54](#)
 - X, [55](#)
 - Y, [55](#)
 - Z, [55](#)
- MpiManager::phdf5_struct, [78](#)
 - halo_max, [78](#)
 - halo_min, [78](#)
 - i_end, [78](#)
 - i_start, [79](#)
 - j_end, [79](#)
 - j_start, [79](#)

- k_end, [79](#)
 - k_start, [79](#)
 - level, [79](#)
 - region, [79](#)
 - writable_data_count, [79](#)
- my_rank
 - MpiManager, [66](#)
- N_lim
 - GridObj, [29](#)
- nProbes
 - definitions.h, [99](#)
- neighbour_coords
 - MpiManager, [66](#)
- neighbour_rank
 - MpiManager, [66](#)
- nu
 - GridObj, [29](#)
- num_ranks
 - MpiManager, [66](#)
- ObjectManager, [67](#)
 - bfl_build_body, [69](#)
 - computeLiftDrag, [69](#)
 - destroyInstance, [70](#)
 - getInstance, [70](#)
 - GridObj, [28](#), [76](#)
 - IBBody, [48](#)
 - IBMarker, [51](#)
 - ibm_apply, [70](#)
 - ibm_banbks, [70](#)
 - ibm_bandec, [71](#)
 - ibm_bicgstab, [71](#)
 - ibm_build_body, [72](#)
 - ibm_computeforce, [72](#)
 - ibm_deltakernel, [72](#)
 - ibm_findepsilon, [73](#)
 - ibm_findsupport, [73](#)
 - ibm_initialise, [73](#)
 - ibm_interpol, [73](#)
 - ibm_jacowire, [73](#)
 - ibm_move_bodies, [74](#)
 - ibm_position_update, [74](#)
 - ibm_position_update_grp, [74](#)
 - ibm_spread, [74](#)
 - io_readInCloud, [75](#)
 - io_restart, [75](#)
 - io_vtk_IBwriter, [75](#)
 - io_write_body_pos, [75](#)
 - io_write_lift_drag, [75](#)
 - io_writeForceOnObject, [76](#)
- ObjectManager.cpp, [109](#)
- ObjectManager.h, [109](#)
 - eBBBCloud, [109](#)
 - eBFLCloud, [109](#)
 - eIBBCloud, [109](#)
 - eObjectType, [109](#)
- ObjectManager_init_bflbody.cpp, [110](#)
- ObjectManager_init_ibmbody.cpp, [110](#)
- ObjectManager_ops_ibm.cpp, [110](#)
- ObjectManager_ops_ibmflex.cpp, [110](#)
 - SWAP, [110](#)
 - TINY, [111](#)
- ObjectManager_ops_io.cpp, [111](#)
- omega
 - GridObj, [29](#)
- onespace
 - GridUtils, [41](#)
- operator()
 - IVector, [53](#), [54](#)
- p_data
 - MpiManager, [66](#)
- PCpts, [76](#)
 - ~PCpts, [77](#)
 - PCpts, [77](#)
 - x, [77](#)
 - y, [77](#)
 - z, [77](#)
- PCpts.h, [111](#)
- path_str
 - GridUtils, [45](#)
- position
 - Marker, [57](#)
- position_old
 - IBMarker, [52](#)
- Q
 - BFLBody, [11](#)
- range_i_left
 - MpiManager.h, [108](#)
- range_i_right
 - MpiManager.h, [108](#)
- range_j_down
 - MpiManager.h, [108](#)
- range_j_up
 - MpiManager.h, [108](#)
- range_k_back
 - MpiManager.h, [108](#)
- range_k_front
 - MpiManager.h, [109](#)
- recv_layer_pos
 - MpiManager, [66](#)
- recv_stat
 - MpiManager, [66](#)
- RefXend
 - definitions.h, [99](#)
- RefXstart
 - definitions.h, [99](#)
- RefYend
 - definitions.h, [99](#)
- RefYstart
 - definitions.h, [99](#)
- RefZend
 - definitions.h, [99](#)
- RefZstart
 - definitions.h, [99](#)

- region
 - MpiManager::buffer_struct, [17](#)
 - MpiManager::phdf5_struct, [79](#)
- SWAP
 - ObjectManager_ops_ibmflex.cpp, [110](#)
- send_requests
 - MpiManager, [66](#)
- send_stat
 - MpiManager, [66](#)
- sender_layer_pos
 - MpiManager, [67](#)
- size
 - MpiManager::buffer_struct, [17](#)
- spacing
 - Body, [16](#)
- stdafx.cpp, [111](#)
 - c, [112](#)
 - cs, [112](#)
 - w, [112](#)
- stdafx.h, [112](#)
 - c, [113](#)
 - cs, [113](#)
 - L_IS_NAN, [113](#)
 - LUMA_FAILED, [113](#)
 - w, [113](#)
- stridedCopy
 - GridUtils, [41](#)
- subGrid_comm
 - MpiManager, [67](#)
- subtract
 - GridUtils, [41](#)
- supp_i
 - Marker, [57](#)
- supp_j
 - Marker, [57](#)
- supp_k
 - Marker, [57](#)
- support_rank
 - Marker, [57](#)
- t
 - GridObj, [29](#)
- TINY
 - ObjectManager_ops_ibmflex.cpp, [111](#)
- tension
 - IBBody, [49](#)
- timeav_mpi_overhead
 - GridObj, [29](#)
- timeav_timestep
 - GridObj, [29](#)
- upToZero
 - GridUtils, [42](#)
- vecmultiply
 - GridUtils, [42](#)
- vecnorm
 - GridUtils, [42–44](#)
- w
 - stdafx.cpp, [112](#)
 - stdafx.h, [113](#)
- world_comm
 - MpiManager, [67](#)
- writable_data_count
 - MpiManager::phdf5_struct, [79](#)
- X
 - MpiManager::layer_edges, [55](#)
- x
 - MarkerData, [59](#)
 - PCpts, [77](#)
- XInd
 - GridObj, [29](#)
- XPos
 - GridObj, [29](#)
- xProbeLims
 - definitions.h, [99](#)
- Y
 - MpiManager::layer_edges, [55](#)
- y
 - MarkerData, [59](#)
 - PCpts, [77](#)
- YInd
 - GridObj, [29](#)
- YPos
 - GridObj, [29](#)
- yProbeLims
 - definitions.h, [99](#)
- Z
 - MpiManager::layer_edges, [55](#)
- z
 - MarkerData, [59](#)
 - PCpts, [77](#)
- ZInd
 - GridObj, [30](#)
- ZPos
 - GridObj, [30](#)
- zProbeLims
 - definitions.h, [99](#)